Few-Shot Learning with Background Subtraction

A Thesis

by

HUI WANG

BS, Texas A&M University-Corpus Christi, 2017

Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

Texas A&M University-Corpus Christi
Corpus Christi, Texas

December 2020

December 2020

Few-Shot Learning with Background Subtraction

A Thesis

by

HUI WANG

This thesis meets the standards for scope and quality of
Texas A&M University-Corpus Christi and is hereby approved.

Longzhuang Li, PhD
Chair

Ning Zhang, PhD
Committee Member

Dulal Kar, PhD
Committee Member

December 2020

ABSTRACT

Few-shot learning for image classification aims to classify the image by only using few images as supporting samples. In the past several years, few-shot learning has achieved a huge improvement in image classification. In the recent work, such as meta-transferlearning (MTL) and Few-shot Adaptive Faster R-CNN have achieved a higher accuracy. In this paper, we are trying to combine three different methods together which are YOLOV2 model, Mask RCNN and our few-shot learning model. When a CNN wants to recognize animals in photos, there is a huge chance that even features that are supposed to represent trees will be encoded as belonging to those animals.Our main idea is to using YOLO algorithm, Mask RCNN and Opencv functions to reduce the noise and background as much as possible and keep our main object as it is. We would like to train our model using the image that only contain the object itself. We show that this approach is helpful to improve the accuracy in few-shot learning image classification.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES PAGE

CHAPTER I: INTRODUCTION

Neural networks have been around since the 1950s, but we just never had as much data and computing power as we do now. The deep neural networks have achieved big success in recent years. Many approaches have been introduced and rustling a good accuracy. However, deep learning relies on a large amount of labeled data and many iterations to train the data in image classification. So, it is really time consuming. On the other hand, due to the lack of dataset, the accuracy could be dropped down. Humans have ability to recognize a new class by looking at only a few or even one image. For example, children can generalize panda from one single picture or hearing the description of panda from others[29]. This is how humans brain works. It is different than the traditional deep learning machine which need to train large number of labeled images to recognize new class. Few shot learning works similar as human brain from this perspective. Few-shot learning aim to solve the problem with small amount of data. Some of the classes have a small amount data. For example, rare animals, secret modern weapons and so on. These are the places where few-shot learning focus on. Many approaches have been introduced recently. Match network, Siamese network, prototype network, optimization based have been discussed in many papers and leading to a better result. The n-shot, k-way classification task has been used in many recent papers. We obtain n classes from the dataset, and each class contain n sample images. So, there will be total n*k samples as our support set. Within the same classes, randomly select the remain picture as the batch set. Our goal is to decide which of the class the unseen batch set belong to by training the support set. In the past few years, many approaches have been presented for example, prototypical network, Optimization Based, Matching Networks and Metric Based. Also, other novel approaches have been introduced recently, they are relied on or highly related to those approaches.

The training image is split into two regions, foreground and background, and they are trained by using two separate convolutional neural networks. The classification is made base on both

convolutional neural networks. As the experiment result shows, the accuracy increased by 2 points with Bath Folding, 4 points with Localization and 5 points with Covariance Pooling on the meta-iNat dataset. Hilliard[6] proposed a metric learning method with conditional embeddings for few host learning. The network is made up of four convolutional blocks where each block begins with a 2D convolutional layer with a 3×3 kernel and filter size of 32. Each convolutional layer is followed by batch normalization, an ELU activation [3], and a 2×2 max pooling layers. After these four convolutional blocks, it produces a vector of size 800 to present the image. The same parameters are used for all images in each few-shit trail, no matter the image is a query image or is from support set. Then the relational network combines information across images in a class[23]. Within these exiting approach, most of them treat the whole image as the input of the convolutional neural network which may encode the noise as the feature of the target image. If the noise is learnt as the target object feature, obviously it has negative impact image classification accuracy. In this paper, we present a combination of YOLOV2, Mask RCNN and few-shot learning DeepEMD network to deal with the noise object or background in order to improve the classification accuracy. YOLOV2(You only look once) is a real time object detection which is widely used to determine the object location and classification. It's a faster algorithm compare with others for example RCNN, Faster RCNN etc. We apply YOLOV2 into our model to identify the object we are interested in, so that we can ignore other noise object or background. Not only that, we also used Mask RCNN to remove the noise inside the bounding box which is produced by YOLOV2 algorithm. Our goal is to remove noise as much as possible from the image and remain the main object before we feed it to our few-shot learning model.

CHAPTER II: REVIEW OF THE LITERATURE

As we all know, deep learning is a very important milestone in the development of machine learning, and deep learning has achieved great success on many tasks. However, because the deep model contains many parameters, it usually requires a large amount of labeled data for model training, which severely limits its application: in many scenarios, collecting large amounts of labeled data is very expensive and difficult , Or even impossible, such as medical data, data manually marked by users on mobile phones, etc. Is it possible to train a good model using only a small amount of labeled data? This has become a very important subject in the development of machine learning, and it is highly concerned by both academia and industry. Few-shot learning refers to learning a new class with only a few examples. In recent years, many approaches are posted with great ideas. There are four basic methods are widely used in few-shot learning field: Metric learning methods, Meta-learning methods, Semantics-based Methods and Optimization Based Methods.

Metric learning methods, where the image is embedded into metric spaces and the feature of the object are learned. The same category are closed and the different category are far away. Davis and Bharath points[34] out that traditional image recognition model requires many, equally large balanced, and labeled classes, but when it comes to real world problem most likely it tends to be heavy-tailed. So it fails when the date set is heavily imbalanced and the object becomes overlapping, tiny, occluded or blurry. They designed a novel framework to deal with this issue. The representation set with many examples, annotated withe bounding boxes are used to train the model. Only varying amounts, very few bounding boxes in the reference images are learned and pass to the classifier. Davis and Bharath points[34]'s approaches, it develops a category-agnostic "foregroundness" model on the representation set to deal with the problem which the object of interest is small or the scene cluttered, since it is unclear what part of the image the label refers to. The classifier depend on prototypical networks. Li[11] proposes a Deep Nearest Neighbor

Neural Network using a CNN-based embedding module for learning deep local descriptors and an image-to-class module for measuring the similarity between a given query image and each of the classes. Features are extracted with a CNN and everything is learned end-to-end. Lifchitz, Yann and Avrithis[13] states that instead of global average pooling at the end all spatial locations are required to be correctly classified. in addition, at the test time, they widen each layer by adding neurons and fine tune them instead of fine-tuning the last layer. Only the additional weights are trained, the old ones are frozen.

Meta learning methodsis also called learning to learn. These are models which are conditioned on the current task, so a different classifier is used as a function of the support-set. The idea is to find model hyper-parameters and parameters such that it will be easy to adapt to a new task without over-fitting to the few shots available. A traditional machine leaning model need a large number of samples to learn its features in order to recognize a new unseen object. Meta learning model aims to solve the problem of lacking samples. It gives machine ability to generalize the new task or new environments that have never been encountered during training time. Few shot learning is the filed that meta learning can be used frequently. With a few of samples, meta learning is much faster and more efficiently compare with a traditional machine leaning model. Sun[27]proposed a novel meta learning method called meta-transfer learning (MTL) which takes the advantages of transfer and meta learning. In this method, a large-scale data is learned from the deep neural network (DNN) for example, 64-class and 600-shot. Then the meta-transfer learning phase learns the parameter of scaling and shifting based on the pre-trained feature extractor. It reduces the number of learning parameters and avoid overfitting problems. In addition, the trained parameters are frozen and only a scale and shift per layer are learned. Meta-test is done for an unseen task which consists of a base-learner (classifier) Fine-Tuning (FT) stage and a final evaluation stage. Lee, Kwonjoon and Maji[9] clam that backbone coupled with an SVM classifier trained end-to-end can result a good accuracy. In many research paper, author tried to reduce the feature dimensions before the classifier, but in this paper, Lee et al the feature dimensions is kept quite high

because the SVM classifier can handle this high dimension. Graph Neural Network[8] has been applied before for few-shot learning.The basic idea is that each image can be presented as a node in a graph and the information (node representation) can be propagated between them according to how similar they are. Gidaris et al.[4] built a meta-model to predict the classifier weights for the unseen classes. Both base and novel classes are going through the a denoising auto-encoder which is implemented as a graph neural network. It allows to propagate knowledge from the base classes classifiers to the novel ones. Denoising auto-encoder can help to fix the predicted classifiers which are predicted based on only a few examples and are obviously noisy.

Semantics-based methods are on the rise. It is inspired by zero-shot learning where classification is done based solely on the category name, textual descriptions, or attributes. Those extra semantic ques can also be of help when visual examples are scarce. Schwartz et al.[25] built their model based on AM3 model, and it generalize it to use multiple semantics. They begin with the visual prototypes and iteratively update them with a sequence of semantic embeddings, and it achieved a very good results on miniImageNet. In many recent papers, researchers has shown that pre-trained concepts are important in order to increase the accuracy[34][33][7]. Schonfeld et. al[24] used two VAEs, one for visual features and the-other one for semantic features. The objective is to be able to reconstruct semantic features from the latent of visual features and vice versa. The work from Wang et. al[33], the label embeddings (GloVe) are used to predict the weights of the visual feature extractor model. They suggest a nice trick of factorizing the weights so only a lower dimension weight vector needs to be predicted. In addition, alignment between the semantic embedding and visual embedding is forced through the "embedding loss". This paper is interesting since it combines two approaches — meta-learning (predicting the model based on task) and using semantic information (labels).

Prototypical network embeds images into metric space and make an assumption that is there exists an embedding in which points cluster around a single prototype representation for each class. The mean of the individual sample has been used in this paper. Then the problem becomes

finding the closest distance from the query image prototype to support set. Prototype network compute prototype for each class. Each prototype is the mean vector of the embedded support points belonging to its class. Vinyals et.al[31] using cosine distance apply to matching network. Snell states that using squared Euclidean distance can greatly improve result for both matching network and prototypical network. The authors used man/prototype within a learned embedding space to represent each class, and the new class has been recognized by calculating the distance to the prototypes. The model is trained by randomly sampling classes and instances per episode. Some insights are given about the connection to mixture density estimation in the case of Bregman divergence-based distances, linear models, and matching networks. The same framework extends relatively straightforwardly to zero-shot learning by making the class prototype be the result of a learned mapping from meta-data like attributes to the prototype vector.

Optimization Based. First of all, these gradient optimization algorithms including momentum, adagrad, adadelta, ADAM, etc., cannot be optimized in a few steps, especially on non-convex problems, the selection of multiple hyperparameters cannot guarantee the speed of convergence.Second, the random initialization of different tasks will affect the task convergence to a good solution. Although the transfer learning of finetune can alleviate this problem, the performance of transfer learning will be greatly reduced when the new data has a large deviation from the original data. We need a systematic general initialization of learning, so that training starts from a good point. Unlike transfer learning, it can guarantee that this initialization can make finetune start from a good point. The model learns about an update function or update rule for model parameters. Instead of learning a single model in multiple rounds of episodes, it learns a specific model in each episode. Specifically, learn the parameter update algorithm based on gradient descent, use LSTM to express the meta learner, use its state to express the update of the target classifier parameters, and finally learn how to initialize the classifier network (learner) on the new classification task. Parameter update. This optimization algorithm considers both short-term knowledge of a task and long-term knowledge across multiple tasks.

CHAPTER III: METHODOLOGY

In this work, we designed a novel architecture for few-show object detection. It requires base class which contains large amount annotated novel class which has limit a few labeled samples. The aim of our model is to recognize the novel class by pre-training base class and only a few support class training. One of the difficulties of few-shot object detection is to extract the main object from the image and learn its features only. We know that animals are usually found near things like trees, grass and soil. Thus if we train a CNN naively on such data then there is a huge chance that even features that are supposed to represent trees will be encoded as belonging to those animals. Images with noise background or fuzzy image even make the problem harder, and it reduce the accuracy rapidly. So how to successfully identity the main object location and remove other noise from image before feed it into training model become important and it has a significant impact on classification accuracy. Firstly our mode use Yolo algorithm [18], one stage network to extract the object from the image. Yolo takes a completely different approach compare with RCNN [5]. RCNN uses selective search to extract 2000 region proposals which is time-consuming. Differently, Yolo is a single convolutional network to extract the bounding boxes and classify the class of these boxes. In this work, our novel framework for few-shot object detection is built on Yolo architecture. Yolo divides up the image into a grid of 13x13 cells. Each of the cell is responsible for predicting five bounding boxes. A bounding box describes the rectangle that encloses an object. The bounding boxes with a confidence score encloses an object. Yolo also outputs a confidence score of these bounding boxes that tells us how certain it is that the predicted bounding boxes actually enclose some object. This score does not say anything about what kind of object is in the box, just if the shape of the box is any good. By setting the threshold we can remove all the useless bounding boxes and remain the the significant ones. The predicted bounding boxes may look something like as the shown. The higher the confidence score, the fatter the box

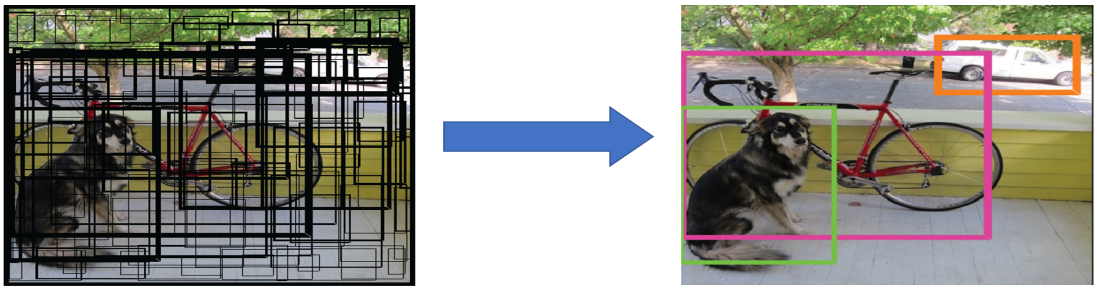

Figure 3.1
The confidence score for the bounding box and the class prediction are combined into one final score that tell the probability that this bouding box contains a an object. For example, the big fat box on the left is 85% sure it contains the object.

is drawn. Noticed, for our dog, bike, and car we got very fat or strong boxes, it can tell there is something significant there. By setting up the threshold value, we can ignore other boxes and remain the significant ones.

To downsampling the image, YOLO's convolutional layers factor 32 to an image of 416 to produce the feature map of 13x13. When we move to anchor boxes we also decouple the class prediction mechanism from the spatial location and instead predict class and objectness for every anchor box. Following YOLO, the objectness prediction still predicts the IOU of the ground truth and the proposed box and the class predictions predict the conditional probability of that class given that there is an object. Using anchor boxes we get a small decrease in accuracy. YOLO only predicts 98 boxes per image but with anchor boxes our model predicts more than a thousand. We follow the backbone (DarkNet-19) to implement the meta feature extractor. Before we feed these meta feature maps to our DeepEMD model to train our model, we apply Mask RCNN to the bounding box in order to remove the noise inside the bounding box. Mask RCNN is an approach to detect and delineate each distinct object of interest in an image. Semantic segmentation is the understanding of an image at the pixel level that is we want to assign an object class to each pixel in the image. During the ROI pooling of Mask RCNN, there is a problem of data loss. This involves the applying of pooling usually max polling on a region of interest the bounding box computer

during object detection. Even ROI align is used, the segmentation produced by Mask RCNN still loss some pixels of the object. To Address this problem, we apply Gaussian function and Sobel filters from OpenCv to the images that produced by YOLO. By calculating the edges differences between Mask RCNN and OpenCv, we can retrieve loss data of the image. Both OpenCv functions and Mask RCNN work at pixel level, gaussian function and sobel operator process clear edges of an image. It exactly shows us which pixels of the image contains the object or not. But sobel operator works at gray level, by comparing the difference between the Mask RCNN and sobel operation, we finally can retrieve the losing pixels. We add those pixels back to our Mask RCNN model in order to get as much details as we can for our target object. By combining Mask RCNN and sobel operation we kept much details for our main object. The output of the Mask RCNN and sobel operation as the input of our few-shot learning model. The feature learner learns how to extract meta features from an input image and it generates a meta feature $F \in R^{w \times h \times m}$. It has m feature maps. Our module takes one support image and embed it into class representation to captures global representation and highlight more important and relevant ones to detect the target object. After acquiring class-specific features F_i , we feed them into the prediction module which is similar as Yolo v2. It produces the objectness score, bounding box location $(x,,y,h,w)$, and $P(F_i) = \{O_i, x_i, y_, w_i, c_i\}$. Our model architecture is shown in Figure 3.2.

Given an example of our image processing concepts. We first use YOLO algorithm to identify the bounding boxes of the object in an image. The reason we are using YOLO algorithem instead of others such as SSD, RCNN or fast RCNN is that YOLO is more efficiency and it requires less memory. As YOLO divides the image into a grid of 13x13 cells, 169 cells would be produced. By setting up the threshold, we only keep the significant cells. The final bounding boxes of an image would look like as shown as Figure3.1, as we shown, there are some parts are missing for the object, such as the leg of the dog, the back tier of the bicycle. This leads to a problem that in some cases, the bounding boxes does not perfectly fit our object. This also happens when we try to remove the noise inside the bounding box using Mask RCNN, but not every image has this kind

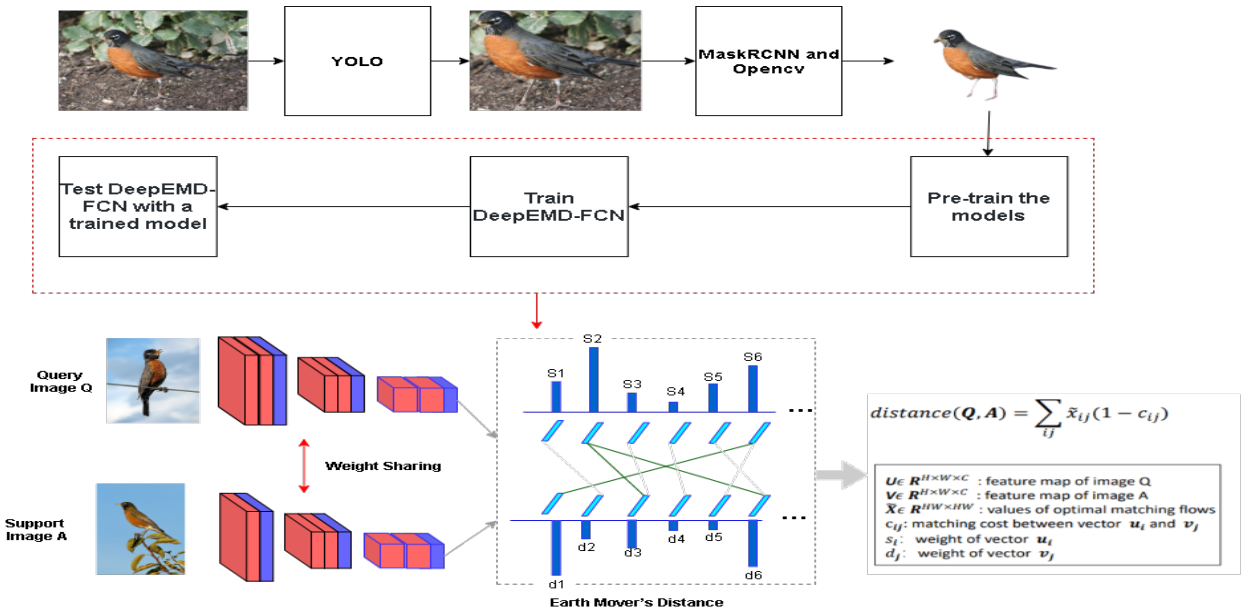

Figure 3.2
The Architecture of our proposed few-shot learning model. It's a combination of Yolo , Mask RCNN, OpenCv, and few-shot learning DeepEMD network. It also contains feature extractor module. The Yolo and Mask RCNN are applied first to deal with the noise of the input image. Yolo algorithm helps to identity the location of the object and produce the bounding box of main object. Mask RCNN works for removing the noise inside the bounding box produced by Yolo, and the outputs as the inputs for our few-shot learning DeepEMD model. The final decision is made through the distance between two object.

of problem, as the Figure 3.3 shows, using YOLO and Mask RCNN can almost perfectly remove the background without losing any information.

While doing the Mask RCNN operation, some part of the image may be missing (For example, see Figure 3.4 and Figure 3.5). Mask RCNN is a neural network to solve the image segmentation problem. In most case it can perfectly seperate the object from the image without losing details. However, for some images when the border of the object has the similar color with the background, the information of the object could be loss. See Figure 3.5.

This missing data could lead us an unsatisfactory result. We try to keep as much details as we can

Figure 3.3
The image shows all the details of the object remained from the Mask RCNN operation.

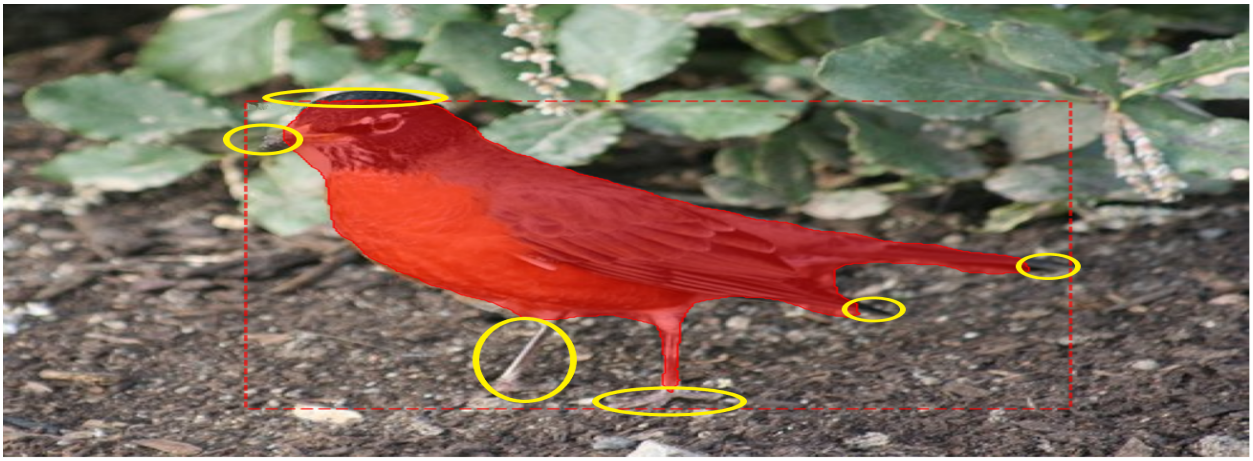

Figure 3.4
The image shows the bird segmentation of the object from the Mask RCNN operation, yellow circle mark the details that are not detected by the Mask RCNN.

while removing the noise from an image. To deal with this problem, we introduce the GaussianBlur function and Sobel filter from OpenCv. GaussianBlur function is used to smoth the background noise, then Sobel filter is able to detect the edge of the object. Sobel filter works at pixel level by calculating the horizontal and vertical pixels differences, it produces the well edges of the object especially for the border of the object because usually the color of the object is fairly different. The

Figure 3.5
The image shows the mouth and legs are losing some part from the Mask RCNN operation.

result using GaussianBlur and Sobel filter is shown as Figure 3.5. The edges contains the most part of details that we are losing from YOLO and Mask RCNN. But those functions only work well in gray pixels level. We would have to add all those pixels back to Our Mask RCNN in order to get much detail as we wish. Because Sobel filter works at gray pixels level, the pixel value tells us the every pixel contains (255) the edge or not (0). So we can easily mark up those pixels and add back to Mask RCNN. This way we can restore our missing details from removing noise inside the bounding box as Figure 3.4 shows. After restoring all the losing details, our result improved evidently. In our early experiment with all details missing, our 5-way-1-shot experiment on Mini-ImageNet accuracy stay at 61.65% wchich is much lower than our final experiment accuracy.

we first created an initial CNN model to overview the performance of CNN implementation in terms of the classification of the object. However, the performance was not satisfactory. Therefore, we further developed the parameter tuning part, added the for loop, and drew all accuracy curve to check the performance of this network.

Earth Mover's Distance (EMD) was first introduced by Yossi Rubner in 2000. It's a distance mea-

Figure 3.6
The image shows the output of the object from the GaussianBlur and Sobel filter operation, the details that losing from Mask RCNN are well remained from this operation, we mark up all the pixels to restore the losing information. After we restore all the losing pixels to our Mask RCNN, we finally produced the image (right), and as input to train our model.

sure between two weighted objects, and now is widely in computer vision filed. We formulated the few-shot learning classification as a matching problem using earth mover's distance. Yossi[21] introduced a transportation problem, suppose that there is a set of sources $S = \{Si|i = 1,2,3.....m\}$ need to be transported to the destinations $D = \{dj|i = 1,2,3.....k\}$. Si represents the supply units of the supplier I, and dj represents the demand of j demander. Cij represents the transportation cost per unit from i to j. The number of units is denoted as Xij. The problem becomes to find the cheapest way to transport all the sources from the suppliers to demanders. In computer vision view, Si and dj are considered as the weight of each node which controls the total matching flows generated by each node. The cost between supplier and demander can be minimized, then the image matching problem can be achieved. Metric based methods in computer image classification aim to calculate the good distance metric and data representations to compare the similarity between two images. Most metric methods compute the distance between images in embeddings level, we decide to use the local information of the images. The image is decomposed into a set of local representations and we use the optimal matching cost between two images to represent

the similarity. We firstly used a fully convolutional network to generate the image embedding $U \in (R^{H \times W \times C})$, where H and W represents the feature maps and C denotes the feature dimension. Each image representation contains a collection of local feature vectors $[u1, u2, ...uHW]$, and each vector u_i can be seen as a node in the set. Thus, the similarity of two images can be represented as the optimal matching cost between two sets of vectors. Following the original EMD formulation, the cost per unit is obtained by computing the pairwise distance between embedding nodes u_i , v_j from two image features where nodes with similar representations tend to generate fewer matching cost between each other. The model generates the weights of all vectors with our proposed cross-reference mechanism Then we use the Earth Mover's Distance to generate the optimal matching flows between two sets that have the minimum overall matching cost. Finally, based on the optimal matching flows and matching costs, we can compute the distance between two images, which are used for classification.

Lose Function. Due to the lack of the data in few shot learning, the lose function can be very important. In this work, we use binary cross-entropy loss function. Our lose as following:

$$\begin{aligned} &min \qquad L := \textstyle\sum_j L(T_j) \\ &\theta_D, \theta_M, \theta_P \end{aligned}$$

$$= \sum_j L_{det}(P_{\theta_P}(D_{\theta_D}(I_q^j) \oplus M_{\theta_M}(S_j)), M_j^q)$$

The function ensure the model learn good meta features for query images. For bounding boxes and objectiveness regression, we also have the similar loss function L_{bbx} and L_{obj} as YOLOV2[19]. Thus, our overall detection loss function is $L_{det} = L_c + L_{bbx} + L_{obj}$. All losses are mean-squared errors, except classification loss, which uses cross-entropy function. Many grid cells do not contain object, the confidence scores of those cells are close to zero. Those cells are significantly affect the gradient of the cell that does contain object. In order to solve this problem, we decrease the

loss from confidence predictions from boxes that do not contain object and increase the loss from bounding box coordinate predictions. Moreover one bounding box only respond to one object. The bounding box represents the object when it has the highest the IOU. The loss function affect the classification only if the the cell present the object.

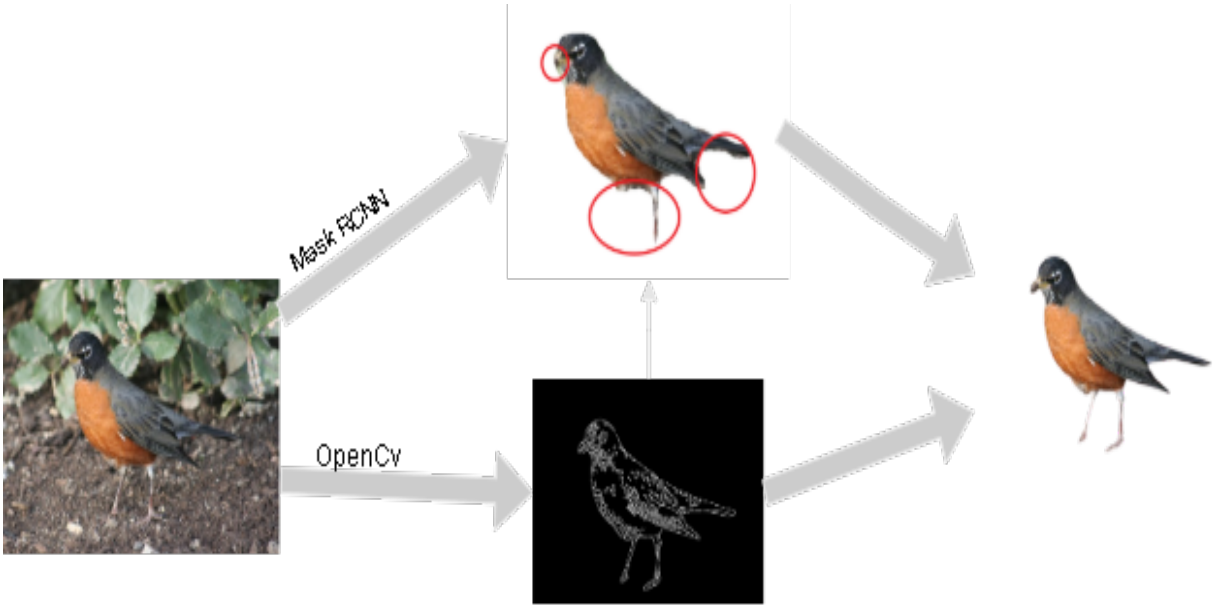

Figure 3.7
To address the data loss of the Mask RCNN, we applied Gaussian, Sobel and bilateral Filters to our images produced by YOLO. By calculating the edges differences between Mask RCNN and Opencv, we can retrieve the losing data during ROI pooling.

Training. There are two steps of training our model, the base class training and base class plus novel class training. On our first training step we have enough data to train our model. Even the base class contains large well annotated data, our feature extractor module is still used. This makes their coordinate in a desired way: the model needs to learn to detect objects of interest by referring to a good reweighting vector. On the second step, we train our model using both base classes and novel classes. We pick the same number of images from the base classes as the novel class has. In order to balance the base class and novel class, we keep k bounding boxes of each base class as same as novel class. Also the training procedure take the same concepts as first phase, the only

difference is that it takes fewer iterations for the model to converge.

Datasets. We evaluate our model for few-shot detection on the widely-used object detection benchmarks, miniImageNet, COCO dataset. We follow the common practice [30, 32, 34, 6]. Out of its 20 object categories, 5 novel classes are select randomly, we keep rest 15 as base classes. The base classes with annotations are trained through base training. Only small set of training images are give to the few-shot fine-turning to guarantee that every class of objects only has k annotated bounding boxes where k equals 1, 2, 3, 5 and 10.

Difference From Other Methods. Our methods is a combination of YOLOV2/YOLOV3, Mask RCNN, OpenCv functions and DeepEMD few-shot learning CNN. Our key idea is to remove as much noise as possible from the original image. By applying the OpenCv functions, we retrieve all the losing pixels from the Mask RCNN and YOLO algorithm. By compare the differences between the OpenCv and Mask RCNN results we are able to keep as much details as we can for our main object from a image. Compare with other few-show learning network, most of them deal with the whole image, this lead to a problem that there is a huge chance that even features that are supposed to represent trees will be encoded as belonging to those animal. We use YOLO algorithm to identify the object that we are interested in. The bounding box of YOLO produced contains the object that we would like to learn the feature from. In some cases, the bounding also contains the other noise object or background. So we use Mask RCNN to remove the noise inside the bounding box to clean out all the noise and remain the main object only. By removing all the noise or background of the image, our model can easily learn the main object features and produce the better accuracy. We developed three additional fully connected layers to train the classification network.For test phase, we use the pre-trained network to extract features from the last fully connected layer.

CHAPTER IV: FINDINGS/RESULTS

Environment. We run our code on Python 3.7.6 and Pytorch 1.4.0. The network is trained on a GTX 2080, 8 GiB with Intel(R) Core(TM) i7-9700K CPU @ 3.60 GHz, 16 GiB memory, and 64-bit OS. **Experiment Result.** We firstly apply YOLOV2 algorithm to the image in order to identify the main object that we interested in. Only one bounding box is produced by YOLOV2 model, by prepossessing the image we only remain the bounding box, other object or background are removed from the image. Then the bounding box image as the input image for our few-shot learning model. We train our model based on DeepEMD[11] for a few-shot learning task in the 5-way and 1-shot setting. For the test phase, we randomly select 600 episodes from the test set, and take the top-1 mean accuracy as the evaluation criterion. The 5-way and 1-shot result is shown. We have our classification results that are higher than most exiting methods, after we successfully combine all three methods together we are expecting a higher accuracy. We also run our code on 5-way and 5-shot task, the accuracy is 74.6%, which is 23.36% higher than 5-way and 1-shot task. As the result we can see that with increase of number of images, the accuracy would have a big jump. Take a whole picture of few-shot learning accuracy, our combination of YOLOV2, Mask RCNN and DeepEMD is still competitive with other methods. Our result is shown as following table:

miniImagenet Dataset			
Method	Backhone	5-way-1shot	5-way-5-shot
cosine classifier[2]	ResNet12	55.43 ± 0.81	77.18 ± 0.61
TADAM[15]	ResNet12	58.50 ± 0.30	76.70 ± 0.30
ECM[17]	ResNet12	59.00 ± -	77.46 ± -
TPN[14]	ResNet12	59.46 ± -	75.65 ± -
PPA[16]	WRN-28	59.60 ± 0.41	73.74 ± 0.19
ProtoNet[26]	ResNet12	60.37 ± 0.83	78.02 ± 0.57
wDAE-GNN[4]	WRN-28	61.07 ± 0.15	76.75 ± 0.11
MTL[28]	ResNet12	61.20 ± 1.80	75.50 ± 0.80
LEO[22]	WRN-28	61.76 ± 0.08	77.59 ± 0.12
DC[12]	ResNet12	62.53 ± 0.19	79.77 ± 0.19
MetaOptNet[9]	ResNet12	62.64 ± 0.82	78.63 ± 0.46
FEAT[35]	ResNet24	62.96 ± 0.20	78.49 ± 0.15
MatchNet[32]	ResNet12	63.08 ± 0.80	75.99 ± 0.60
CTM[10]	ResNet18	64.12 ± 0.82	80.51 ± 0.13
DeepEMD[36]	ResNet12	65.91 ± 0.82	82.41 ± 0.56
Ours[36]	**ResNet12**	**66.81 ± 0.80**	**83.51 ± 0.23**

MS-COCO dataset which contain 80 object categories, 20 categories are novel classes. The remaining 60 categories serve as base class. MS-COCO is well-known and challenging dataset compare with others. We use 50 categories for training, 20 categories for validation and 20 categories for testing. We train our model using 5-way-1-shot, and the result is shown as table below.

MSCOCO Dataset			
Method	Embedding	Type	5-way-1-shot
ProtoMAML[30]	Conv-64F	Metric	41.3 $\pm$ 1.0
CNAPS[20]	Conv-64F	Metric	42.3 $\pm$ 1.0
Squared Euclidean[20]	Conv-64F	Metric	42.9$\pm$ 1.1
AR-CNAPS[14]	Conv-64F	Metric	44.35 $\pm$ 1.1
Simple AR-CNAPS[1]	Conv-64F	Metric	46.2 $\pm$ 1.1
Ours	**Conv-64F**	**Metric**	**45.8 $\pm$ 1.0**

CHAPTER VI: SUMMARY AND CONCLUSIONS

In this paper, we combine the three different methods together in order to remove the noise as much as possible, and remain the main object itself before we feed the image to our few-shot learning model. This way our model can only learn the feature from our target object. Object like tress, grass, river or other noise will be completely removed from the image. This is a good way to keep our model learn the features that are only from the main object.

Future Work. We have combined YOLOV2 model Mask RCNN, Opencv functions and DeepEMD network together to train our model. It leads to a better accuracy compare with other methods. In the future, we would like to try not only using foreground but also background to train our model because in most images, the same objects/animal would have similar environment. For example, birds usually stands on threes, cats images may contains a indoor or out door background. Those backgrounds may be considered as the living feature for an object and become a important part of the image classification. Hopefully this will lead us to a better result. Moreover, we also would like to try other few-shot learning model combining with YOLO and MasK RCNN, for example Dynamic-Netm, MM-Net, and Ralation-Net. Ralation-Net is considered that the measurement method is also a very important part of the network, which needs to be modeled, so the network does not satisfy a single and fixed distance measurement method, but trains a network to learn (such as CNN) the distance measurement method. Loss has also changed. Considering that the relation network pays more attention to relation score, it is more like a regression than 0/1 classification, so MSE is used replaces cross-entropy. We are expecting a improvement of accuracy with our combined method compare with those original methods.

REFERENCES

[1] Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, and Leonid Sigal. Improved few-shot visual classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14493–14502, 2020.

[2] Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

[3] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). arxiv 2015. *arXiv preprint arXiv:1511.07289*.

[4] Spyros Gidaris and Nikos Komodakis. Generating classification weights with gnn denoising autoencoders for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 21–30, 2019.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[6] Nathan Hilliard, Lawrence Phillips, Scott Howland, Artem Yankov, Courtney D Corley, and Nathan O Hodas. Few-shot learning with metric-agnostic conditional embeddings. arxiv. *arXiv preprint arXiv:1802.04376*, 2018.

[7] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[8] Jongmin Kim, Taesup Kim, Sungwoong Kim, and Chang D Yoo. Edge-labeling graph neural network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2019.

[9] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[10] Hongyang Li, David Eigen, Samuel Dodge, Matthew Zeiler, and Xiaogang Wang. Finding task-relevant features for few-shot learning by category traversal. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–10, 2019.

[11] Wenbin Li, Lei Wang, Jinglin Xu, Jing Huo, Yang Gao, and Jiebo Luo. Revisiting local descriptor based image-to-class measure for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7260–7268, 2019.

[12] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[13] Yann Lifchitz, Yannis Avrithis, Sylvaine Picard, and Andrei Bursuc. Dense classification and implanting for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9258–9267, 2019.

[14] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, Eunho Yang, Sung Ju Hwang, and Yi Yang. Learning to propagate labels: Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018.

[15] Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *Advances in Neural Information Processing Systems*, 31:721–731, 2018.

[16] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan L Yuille. Few-shot image recognition by predicting parameters from activations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7229–7238, 2018.

[17] Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Few-shot learning with embedded class models and shot-free meta training. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 331–339, 2019.

[18] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[19] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[20] James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. In *Advances in Neural Information Processing Systems*, pages 7959–7970, 2019.

[21] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000.

[22] Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

[23] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

[24] Edgar Schonfeld, Sayna Ebrahimi, Samarth Sinha, Trevor Darrell, and Zeynep Akata. Generalized zero- and few-shot learning via aligned variational autoencoders. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[25] Eli Schwartz, Leonid Karlinsky, Rogerio Feris, Raja Giryes, and Alex M Bronstein. Baby steps towards few-shot learning with multiple semantics. *arXiv preprint arXiv:1906.01905*, 2019.

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

[27] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[28] Xin Sun, Zhenning Yang, Chi Zhang, Keck-Voon Ling, and Guohao Peng. Conditional gaussian distribution learning for open set recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13480–13489, 2020.

[29] Daniel Swingley. Fast mapping and slow mapping in children's word learning. *Language learning and Development*, 6(3):179–183, 2010.

[30] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, et al. Meta-dataset: A dataset of datasets for learning to learn from few examples. *arXiv preprint arXiv:1903.03096*, 2019.

[31] Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

[32] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[33] Tao Wang, Xiaopeng Zhang, Li Yuan, and Jiashi Feng. Few-shot adaptive faster r-cnn. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[34] Davis Wertheimer and Bharath Hariharan. Few-shot learning with localization in realistic settings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6558–6567, 2019.

[35] Han-Jia Ye, Hexiang Hu, De-Chuan Zhan, and Fei Sha. Learning embedding adaptation for few-shot learning. *arXiv preprint arXiv:1812.03664*, 2018.

[36] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12203–12213, 2020.