Co-Tuning for Transfer Learning on TACO Dataset

Implementation Track, 3 Team members

Abstract

Training a deep neural network from scratch requires huge amounts of computational power, especially in an area like computer vision where the dataset usually contains more than ten thousand images. Moreover, availability of well-defined, large scale, balanced dataset is a prevalent problem in computer vision which leads to sub-par accuracy and a cumbersome pre-processing routines on the dataset. In this paper we re-implemented the breakthrough technique called co-tuning which cascades a neural-network for learning probabilistic relationship on top of a pre-trained backbone network (ResNet50) to improve on the overall object detection accuracy. We take a novel data-set called TACO and attempt to show that the technique of Co-tuning can be implemented on top of transfer learning to improve on object detection accuracy while at the same time keeping the total training time minimal. Co-Tuning relies on the assumption that the pre-trained model is large and diverse enough so that the methodology is generic and can be scaled to multiple types of datasets without any loss of generality.

1 Motivation

Transfer learning is a technique introduced to overcome the tedious and computationally expensive task of learning high number of parameters from the beginning and also providing a concession on the size of dataset required to get a well-trained model with decent detection accuracy. Since Transfer Learning uses pre-trained weights, to learn a good model, smaller dataset are sufficient since the pre-trained model would have already learnt some representation of the image space and only minor modifications might be required. in regards to optimization, initialization of weights of a model using pre-trained is effective. Traditional transfer learning methods involve detaching and discarding the last couple of layers of the pre-trained model and using only the remaining part to perform transfer learning on a newer dataset. These last layers too contain pre-trained weights and formulating a way to use them might speed up the transfer learning process while also making it more robust. You et al. [2020] have tried and successfully tackled this problem.

2 Problem Statement

As described in Section 1, transfer learning has been a hot topic in recent times as scarcity of data for training a deep neural network model is a prevalent problem. To overcome the drawbacks of small datasets, transfer learning utilizes models pre-trained on larger datasets and fine-tunes the pre-trained model by training them on the smaller dataset. In the field of computer vision, numerous models of varied neural architecture are available that have been pre-trained on ImageNet [Deng et al., 2009] such VGG [Simonyan and Zisserman, 2014], ResNet [He et al., 2016] and AlexNet[Krizhevsky et al., 2012]. These pre-trained models are made readily accessible by libraries like PyTorch and TensorFlow. General transfer learning algorithms. The previous works in the field of transfer learning utilize only the bottom layers and discarding the top layers (task-specific layers) of the pre-trained model. Referring to Table 1 provided by You et al. [2020], we can observe that discarding the task-specific layers wastes almost 20% of the total parameters. You et al. [2020] have proposed a method to overcome this drawback of traditional transfer learning and fully transfer the pre-trained deep neural network models.

Pre-trained model	ResNet-50	DenseNet-121	Inception-V3	BERT-base
Task-specific parameters($*10^6$)	2.0	1.0	2.0	22.9
Total parameters($*10^6$)	25.6	8.0	27.2	108.9
Percentage(%)	7.8	12.5	7.4	21.0

Table 1: Pre-trained models parameter count

In general the dataset on which the pre-trained model was trained on and the dataset on which we wish to perform transfer learning, share the same input space but might not have the same marginal distributions. You et al. [2020] have proposed that by finding a probabilistic relationship between the source domain categories and target domain categories, the pre-trained deep neural network model can be used in its entirety. You et al. [2020] have claimed that their proposed Co-Tuning framework improves the accuracy by upto 20%. We would be implementing their approach and verifying the results and also report the performance of Co-Tuning transfer learning on TACO dataset published by Proença and Simões [2020].

3 Related Work

3.1 Pre-trained Models

Computer vision has had a line of trailblazing architectures based on deep neural networks. AlexNet(Krizhevsky et al. [2012]) was a GPU based architecture that surpassed basic feature-engineering based models. ResNet(He et al. [2016]) eased the training of deep neural networks by reformulating layers as "learning residual functions". Researchers have continually used such models pre-trained on these architectures to achieve high accuracy in tasks pertaining to their specific use-cases. More recently, Kornblith et al. [2019] shows that better the pre-trained model, better is the performance of any transfer learning based architecture.

3.2 Domain Adaptation

Domain adaptation aims to solve the problem of classification using a similar idea. The key difference between domain adaptation and transfer learning is that the source domain and the target domain have to be in the same feature space in domain adaptation. This implies that the classification category of both the domains needs to be the same. On the other hand, transfer learning allows for the two domains to be in different category spaces. Moreover, in transfer learning it is not mandatory for the training model to have both the datasets to be available at the time of training while that is not the case with domain adaptation.

3.3 Fine tuning

In fine tuning, the model's output is changed to fit the new task. Fine tuning allows for training only the output layer of the original model. For example, projects in the Application track might discuss relevant works on the dataset, similar analysis. Projects in the Open-ended track should discuss relevant prior approaches and discuss how they are different from and related with your method. Cetinica et al. [2018] explores applying finely tuned CNNs on art classification on 5 different classification tasks belonging to similar classification domain i.e. artwork.

3.4 Continual Learning

Continual learning refers to updating the original model to include new tasks and type of datasets it is able to classify, at the same time, keeping intact its ability to perform good on the original task.You et al. [2020] refers to Kirkpatrick et al. [2017] which implements continual learning based on a methodology called "elastic weight consolidation" which restricts the change in weights after adding new tasks to the original model. You et al. [2020] claims that co-tuning performs better than continual learning when it comes to the paradigm of transfer learning because it is better able to fit the relationship between target and source domains.

4 Method

4.1 Base transfer learning

Let $D_s = \{(x_s^i, y_s^i)\}_{i=1}^{m_s}$ be the source domain and $D_t = \{(x_t^i, y_t^i)\}_{i=1}^{m_t}$ be the target domain where x is the input and y is the output category, hereafter referred to as source domain or target domain category. Let f be the Deep Neural Network trained on the source domain. The traditional transfer learning framework involves splitting f into representation function F_θ parameterized by θ and task-specific function G_{θ_s} parameterized by θ_s. Traditional fine-tuning mechanism for transfer learning only utilizes F_θ and discards G_{θ_s}. Let H_{θ_t} be the function that is introduced to replace G_{θ_s} that outputs in the category space Y_t of target domain. Introducing $l(\cdot)$ as cross-entropy loss we can mathematically express the vanilla transfer learning process as follows:

$$(\theta^*, \theta_t^*) = \underset{\theta, \theta_t}{\arg\min} \frac{1}{|D_t|} \sum_{i=1}^{m_t} l(H_{\theta_t}(F_\theta(x_t^i)), y_t^i)$$

Similar to other machine learning techniques, to prevent the model from over-fitting, regularizers are applied on this vanilla transfer learning formulation. Upon observation, we can see that the transfer learning formulation does not utilize the entire pre-trained model. You et al. [2020] have proposed an algorithm to modify the optimization function such that it is capable of fully transferring the pre-trained deep neural network model.

4.2 Co-tuning methodology

The main idea of co-tuning is to find a probabilistic relationship between the source category space Y_s and the target category space Y_t such as the conditional probability distribution: $p(y_s|y_t)$, by leveraging the fact that G_{θ_s} models the probability distribution over Y_s. Essentially You et al. [2020] have devised an algorithm to map the target domain category labels and the distribution of the source domain category space. After estimating the conditional probability distribution we can alter the transfer learning process by including the task-specific function of the pre-trained model G_{θ_s} as follows:

$$(\theta^*, \theta_t^*, \theta_s^*) = \underset{\theta, \theta_t, \theta_s}{\arg\min} \frac{1}{|D_t|} \sum_{i=1}^{m_t} [l(H_{\theta_t}(F_\theta(x_t^i)), y_t^i) + \lambda l(G_{\theta_s}(F_\theta(x_t^i)), p(y_s|y_t = y_t^i))]$$

The paper proposes two ways to compute the relationship between the source categories and the target categories, a direct approach and a reverse approach. Direct approach involves considering the pre-trained model as a conditional probability distribution over the given input, i.e., $f(x) \approx p(y_s|x)$. Thus by using this notation we can formulate the direct approach as follows:

$$p(y_s|y_t = y) \approx \frac{1}{|D_t^y|} \sum_{(x, y_t) \in D_t^y} f(x), D_t^y = \{(x, y_t) \in D_t | y_t = y\}$$

The reverse approach involves using the pre-trained model f to calculate $p(y_t|y_s)$ from $(f(x_t), y_t)$ and then estimating $p(y_s|y_t)$ from $p(y_t|y_s)$ using Bayes's rule. You et al. [2020] have stated that the direct approach is simple and straightforward but the reverse approach is more effective. The algorithm for the reverse approach is as follows:

To calibrate the deep neural network, You et al. [2020] have used the method *temperature scaling* as stated by Guo et al. [2017]. 2 describes the temperature scaling algorithm for calibration. Guo et al. [2017] have observed that temperature scaling calibration method works surprisingly well for ResNet [He et al., 2016] and DenseNet [Huang et al., 2017] on CIFAR dataset [Krizhevsky, 2009] and ImageNet dataset[Deng et al., 2009]. You et al. [2020] advocate that the pre-trained model is released with its corresponding calibration parameter.

Given that the calibrated pre-trained model is available, it is worth observing that the source domain dataset is not required to estimate $p(y_s|y_t)$. We only rely on the calibrated pre-trained model $\tilde{f}$ and the target domain dataset D_t. Figure 1, taken from You et al. [2020], pictorially depicts the Co-Tuning training process.

Algorithm 1 Reverse approach to learn category relationship $p(y_s|y_t)$

```
Calibrate $f$ according to Alg. 2 using $D_s^v$, where $D_s^v = \{(x_s^i, y_s^i)\}_{i=1}^{m_v}$ is the source validation data.
Let $\tilde{f}$ be the calibrated pre-trained model.
Estimate $\tilde{D}_t = \{(\tilde{f}(x_t^i), y_t^i)\}_{i=1}^{m_t}$ from the the training data $D_t = \{(x_t^i, y_t^i)\}_{i=1}^{m_t}$.
Split $\tilde{D}_t$ into $\tilde{D}_t^{train}$ and $\tilde{D}_t^v$ (validation set).
Train a neural network $g$ on $\tilde{D}_t^{train}$ to learn the mapping from calibrated source predictions to target labels.
Calibrate $g$ according to Alg. 2 using $\tilde{D}_t^v$ and let the calibrated model be $\tilde{g}$.
$\tilde{g}(y_s)$ approximates the probability distribution of $y_t$ conditioned on $y_s$, i.e. $\tilde{g}(y_s) \approx p(y_t|y_s)$
Estimate the marginal probabilities of $p(y_s)$ and $p(y_t)$ from $\tilde{D}_t$.
Estimate $p(y_s|y_t)$ from $p(y_t|y_s)$ using the marginal distributions $p(y_s)$ and $p(y_t)$: $p(y_s|y_t) = \frac{p(y_s)}{p(y_t)} p(y_t|y_s)$
```

Algorithm 2 Neural network calibration

```
Return calibrated function $\tilde{f}(x) = f(x)/t^*$ from $f$, where
$t^* = \arg\min_{t>0} \sum_{i=1}^{m}$ cross_entropy(softmax($f(x^i)/t, y^i$))
```

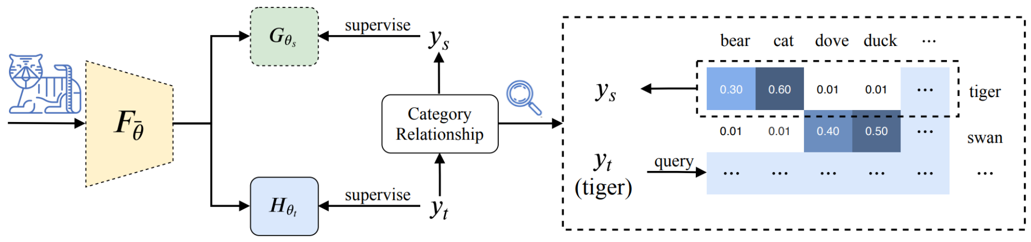


Figure 1: Co-Tuning training pipeline

5 Experiments

TACO dataset [Proença and Simões, 2020] was chosen to check the performance of Co-Tuning algorithm. Similar to You et al. [2020], ResNet50 pre-trained model was used to implement the co-tuning algorithm. We have referred to the code provided by You et al. [2020], for the co-tuning algorithm, to build our code to implement co-tuning on TACO dataset. Therefore there are bound to be some similarities between our codes. We have chosen a novel dataset not provided in the paper since our implementation does not involve coding from scratch. The results for relationship learnt between the TACO dataset and ImageNet dataset have been provided below along with the testing accuracy of the fully transferred deep neural network on the TACO dataset. We tried to get the best accuracy on relationship learning using multiple values of regularization parameter '`C`'. As shown in Figure 2, the best accuracy was achieved for '`C`' = 3 and the value was 52%.

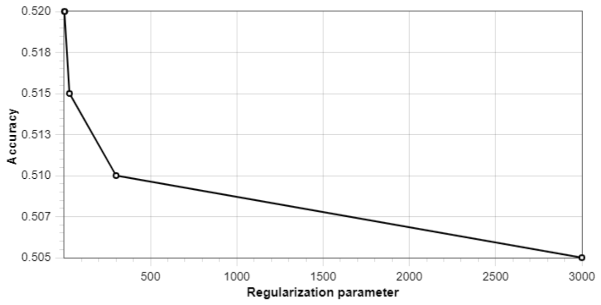


Figure 2: Accuracy vs Regularization parameter

5.1 Dataset

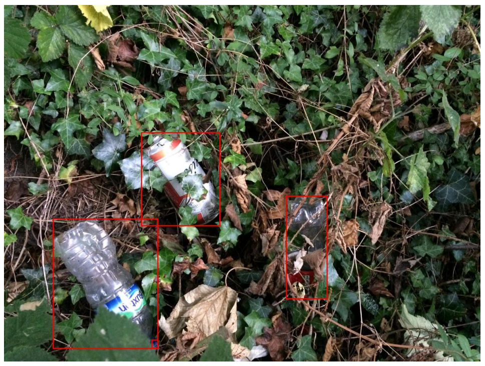
5.1.1 Overview

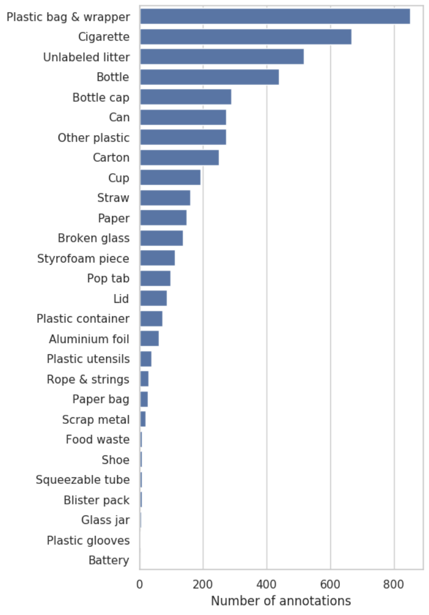
TACO (Trash Annotations in Context) [Proença and Simões, 2020], is an open source image dataset consisting of annotated assorted varieties of waste in the wild. TACO dataset has been used for testing the performance of Co-Tuning for the following reasons

- TACO is relatively new dataset. It is a novel dataset in the sense that not many models have been developed and tested on it. This provides us the opportunity to explore more.
- You et al. [2020] have stated that co-tuning works best on small to medium scale datasets. The dataset consists of 1500 images spread across 60 categories such as cigarettes, bottles, cans, etc.
- This work on TACO dataset can be further extended to classification of recyclable and non-recyclable wastes, which could help with waste segregation and categorization of litter at waste management facilities. We believe that this work can contribute towards fixing this ever worsening problem.

5.1.2 Challenges

Since much research has not been done on TACO dataset we were faced with a lot challenges and this section explains a few of them and the solutions we employed to try and overcome those challenges.


(a) Single Image with multiple bounding boxes



(b) Super Categories in TACO dataset

Figure 3: The TACO Dataset

Most of the images in the TACO dataset contain multiple object(waste) in a single image as shown in Figure 3a. Instead of working with multi-class multi label classification problem, we decided to use the bounding box information in the annotation data to crop out the objects from images with multiple objects and use them as separate images for training. Since TACO is relatively new dataset and since we had to custom modify the images as mentioned above, a significant portion of effort was focused on writing the `dataloader` function from scratch.

The dataset has ~30 categories which were highly imbalanced. As seen in Figure 3b, categories like cigarette and plastic bottles have high number of images in them. This was natural since most of the trash comes from plastic wastes. The dataset contained categories like "unlabelled litter" which was ambiguous from object detection perspective. A number of categories strangely have few to no images in them which was specially tricky for transfer learning since SKlearn's implementation of logistic regression discards any category with no image in it. Since dataloading was random, we frequently missed images from some of these categories, and even when we didn't, some of those categories had too few examples to be effectively used as training examples. To overcome this obstacle we chose categories that were sufficiently defined, i.e., we chose the categories that had a good enough number of images associated to them. We also discarded ambiguous categories like "unlabelled litter".

5.2 Training

Pre-trained ResNet50 model readily available in `torchvision.models` was used in our implementation. The first 50 convolution layers were considered as F_θ and the last fully connected layer was considered as G_{θ_s}, where F_θ and G_{θ_s} are representation function and task-specific function respectively as defined in Section 4. In the code, F_θ and G_{θ_s} have been defined as `ResNet50_F` and `ResNet50_C` respectively. Figure 4 shows the processes involved in co-tuning a pre-trained model.

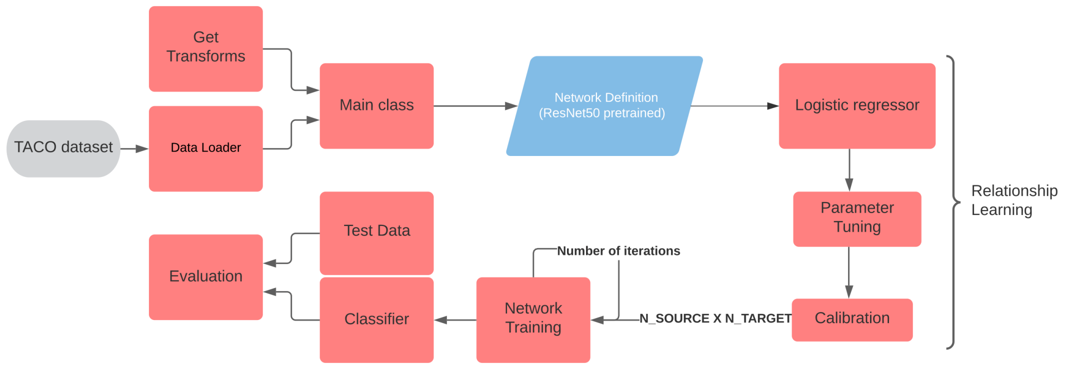


Figure 4: Algorithm Flowchart

5.2.1 Data Pre-processing

As mentioned in Section 5.1.2, we had to write the `dataloader` function from scratch. You et al. [2020] have used multiple image transforms while loading their dataset but since TACO is new and was challenging to use as mentioned in Section 5.1.2, we had to carefully analyze the transforms used by You et al. [2020] and decided to use the following transforms: `torchvisions.transforms.Resize()` `torchvision.transforms.functional.crop()`, `torchvision.transforms.normalize()` and `torch.type(FloatTensor)`. Despite the original paper using random cropping, we decided not to use `torchvision.transforms.RandomCrop()` as the original dataset is already annotated with bounding boxes, so all the relevant information is present over the entire image.

As explained in Section 5.1.2, the TACO dataset has multiple images with more than one object in a single image. Thus as a pre-processing step, we decided to extract out the cropped image from bounding box information provided in the annotated dataset. Furthermore, Since the bounding box could vary in shape/size but the networks demands the image to be uniformly sized, we used `torchvisions.transforms.Resize()` to resize the image into a fixed 256×256 size input. Since we used to NumPy to load the images, the pixel values were integers in the range of 0 to 255, therefore normalized each channel(R,G,B) between 0 to 1. The data is mean-normalised so as to be mitigate the problems of a highly imbalanced dataset. Furthermore, we wrote from scratch a horizontal flip transform so that the data could be as randomised as possible.

The final dataset that we created via extracting out individual images corresponding to each bounding box contained 4000 images. This helped us in 2 ways:

- Increased the size of dataset enabling us to better train the model. This worked as a type of dataset augmentation.

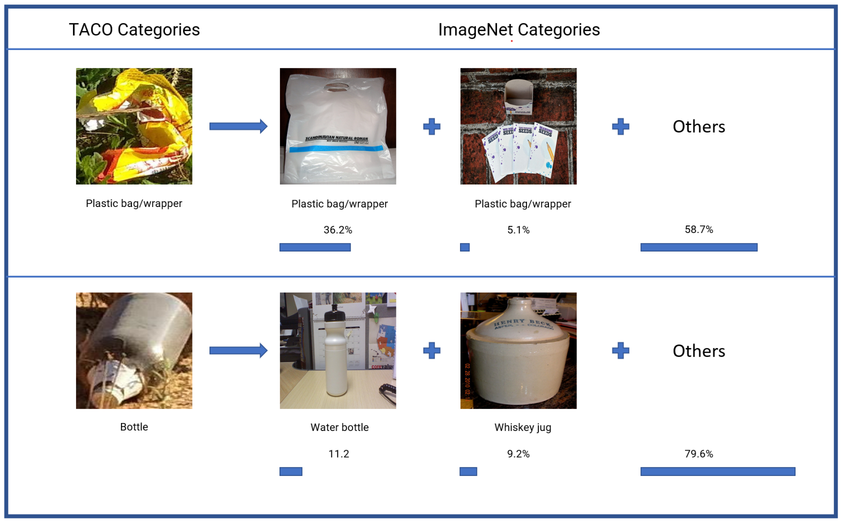


Figure 5: Relationship learnt between TACO dataset and ImageNet dataset

- Cropped images from bounding boxes have content-to-noise ratio thereby helping making the model learn better representations.

The overall dataset, after extracting bounding boxes from each image, was split into train, validation and test sections, containing 3800, 200 and 113 images respectively, with the order of selection being random.

5.2.2 Relationship training

A logistic regression classifier was implemented to model the relationship between the source domain and the target domain. To find the best classifier parameter, we run an iterative loop on multiple values of regularization parameter '`C`'. We find the best regularization parameter and use that for our target category prediction. The model g as mentioned in Alg. 1 has been implemented using SKlearn's implementation of classical logistic regression trying to map the source logits $G_{\theta_s}(F_\theta(x_t^i))$ to target labels y_t^i. Thus the model g has the logistic regression loss function as follows:

$$\arg\min_{w,b} \frac{1}{2} w^T w + C \sum_{i=1}^{m_t} \log(\exp(-y_t^i (G_{\theta_s}(F_\theta(x_t^i))^T w + b)) + 1)$$

6 Results

6.1 Learnt Relationship

The learnt relationship between the source domain labels and the target domain labels is in the form of a probability distribution $p(y_s|y_t)$. We visualize the learned relationship between the target domain labels (TACO dataset) and the source domain (ImageNet) labels between 600 and 800 in the Figure 6. This matrix was used to convert a label in the TACO dataset to a probability distribution over the ImageNet dataset, which was subsequently used as a target vector for training the ResNet50 classifier layer. Figure 5 shows the probability distribution of TACO dataset super categories "plastic bag/wrapper" and "bottle" in the category space of ImageNet. We can observe that Alg. 1 is able to capture the context of target categories allowing it map the target categories onto the source categories. Hence we have verified the relationship learning aspect of the Co-Tuning algorithm proposed by You et al. [2020].

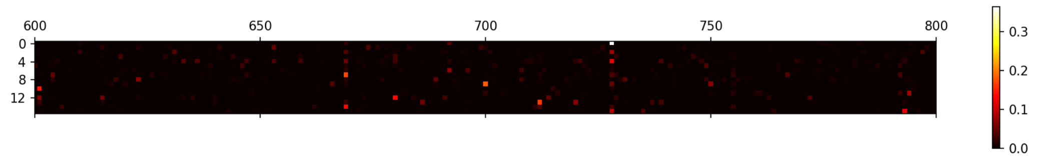


Figure 6: Relationship matrix visualization

6.2 Evaluation

First we evaluated the accuracy of the logistic regression model which predicted the source labels from the target labels, which was found to be 58% on the training dataset and 52% on the testing dataset. This relationship was computed without seeing any source domain data, and the relationship was solely based on the information about the source dataset retained in the pre-trained ResNet model. Finding this accuracy satisfactory, we moved on to training the co-tuning model which involved the novel loss.

We were successful in modelling the relationship to a high accuracy given how imbalanced the data was. Our expectation for correct relationship matching was from 40-50% given that a large number of images belonged to some fixed number of categories. The final accuracy was a bit better than what we expected. Further, as the next step we evaluated the accuracy of our complete model on a training set of size 113, which was randomly sampled at the start of our training experiment. The accuracy that we obtained with our network was 20.1%. We investigated into the possible reasons for the low accuracy of the model, and found that the gradients of the model were stuck at zero due to the presence of `NaN` values in the training flow. This could possibly be due to the high imbalance in the target domain dataset, and also due to high variation in the training images for the same supercategory. The training process can also benefit from being subjected to a broader range of hyperparameter tuning range to investigate their effect on the training quality.

7 Conclusion

Co-tuning is a game changing technique given how mature deep neural networks are getting with every passing day. Its usually quite cumbersome to model the network from scratch for a specific task. With large networks (such as the ResNet50) which are already quite mature and accurate, transfer learning makes practical sense in more applications than not specially in vision tasks like object detection. Thus, novel research on improving object detection accuracy via transfer learning is an active area of research where accuracy improvements are possible. Co-tuning is a step towards that, and the novel implementation of probabilistic relationship generation already boasts good improvements in detection accuracy on medium to large scale datasets. While we were unable to achieve a good accuracy on the classification task for the TACO dataset, we were able to learn a fairly useful and accurate relationship between the target dataset (TACO dataset) and the source (ImageNet) dataset. Our results show that it is possible to augment low amounts of data effectively through relationship learning. This also serves as evidence to the fact that the last layers of a large pre-trained network can still be useful for establishing relationships with smaller datasets, indicating their retention of information about the original dataset.

8 Acknowledgement

We would like to sincerely thank our course instructors. We were in face with multiple bottlenecks that were pointed out during course lecture and assignment problems. The practicality of problem solving taught by professor Clayton Scott and Professor Alfred Hero helped us during the course of our project. This project truly made us go beyond the scope of this course and research new areas in deep learning, probabilistic modelling and transfer learning. We also thank course's GSIs for continually helping us during the course of this project right from picking up the topic to understanding the logic and implementing it. We are also grateful to You et al. [2020] for making their code repository available for reference, which helped us build our implementation of the algorithm.

9 Contributions

- *Literature survey* A1, A2 and A3 carried out the literature survey on the methods used related to transfer learning jointly. A1 read the papers Kirkpatrick et al. [2017] and Huang et al. [2017] to survey the latest techniques in continual learning. A2 and A3 studied the latest research work in Cetinica et al. [2018] to explore other techniques as well. A1, A2 and A3 narrowed down on co-tuning as the approach to implement.
- *Proposal Writing* Proposal writing was done jointly by A1, A2 and A3. A1 wrote the related works section after a knowledge transfer between the three of us. A3 was responsible for methods sections, milestone setting and scheduling tasks.
- *GitHub* A1 created and maintained the GitHub repository and laid down the coding and committing ethics related to our code.
- *Dataset Loader Implementation* A1 and A2 wrote the loader logic pertaining to the TACO dataset. A2 came up with the idea of carving out the bounding boxes as individual images and use them as the training inputs. A2 wrote the implementation for extracting out supercategories and use those super categories instead of actual categories for classification. A1 implemented the data transforms used for pre-processing the data (data resize, data crop etc.) A2 worked on improving the dataloader implementation for better performance, modularity and introduced methods to make it easier to debug.
- *Checkpoint mechanism* A1 wrote the module for model check-pointing and restoring. To avoid multiple training cycles during testing of our implementation, it was required to write a check-pointing mechanism to save on time.
- *Relationship Learning Implementation* A2 worked to understand and simplify the crux of relationship learning and developed a plan of approach to code the network. A1 and A2 wrote the logic for the networks that learns the relationship between source categories and target categories.
- *Backbone Training* A3 wrote the backbone ResNet architecture. A3 came up with the study of how many layers of the ResNet50 backbone network need to be retrained.
- *Experimentation* The final experimentation on how to improve on the accuracy of overall model was done jointly by A1, A2 and A3. A3 resolved one bottleneck when the model was malfunctioning by not normalizing the image pre-processing layer to generate floats in the range $[0, 1]$.
- *Final Project Report* A1, A2 and A3 jointly created the final project report. A1 plotted the hyperparameter tuning curve for relationship training, and together with A3 created the flowchart model of the code. A3 created the visual interpretation of the relationship between the TACO and the ImageNet dataset, while A2 captured the numerical qualities of the matrix in the matrix visualization. Everyone worked on editing the final report.

References

Eva Cetinica, Tomislav Lipica, and Sonja Grgic. Fine-tuning convolutional neural networks for fine art classification. *Expert Systems with Applications*, 114, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

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

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2661–2671, 2019.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

Pedro F Proença and Pedro Simões. Taco: Trash annotations in context for litter detection. *arXiv preprint arXiv:2003.06975*, 2020.

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

Kaichao You, Zhi Kou, Mingsheng Long, and Jianmin Wang. Co-tuning for transfer learning. *Advances in Neural Information Processing Systems*, 33, 2020.