WHAT HAPPENS TO THE SOURCE DOMAIN IN TRANSFER LEARNING?

Amal Alnouri
Damascus University
alnouri.amal@gmail.com

Bilal Alsallakh
Voxel AI
bilal@voxelai.com

ABSTRACT

We investigate the impact of the source domain in supervised transfer learning, focusing on image classification. In particular, we aim to assess to which extent a fine-tuned model can still recognize the classes of the source domain. Furthermore, we want to understand how this ability impacts the target domain. We demonstrate how the retained knowledge about the old classes in a popular foundational model can interfere with the model's ability to learn and recognize the new classes. This interference can incur significant implications and highlights an inherent shortcoming of supervised transfer learning.

1 INTRODUCTION

Transfer learning (TL) has played a substantial role in successful application of deep neural networks in data-lacking domains such as medical diagnosis (Das et al., 2020; Jaiswal et al., 2021) and autonomous driving (Sumanth et al., 2022). The prevalent TL paradigm in computer vision and image processing is to fine-tune a foundation model on the downstream task. Foundation models are able to recognize various visual features and are typically pre-trained on large-scale image datasets such as ImageNet. Pre-trained models are usually available for popular architectures such as ResNet (He et al., 2016) and VGGNet (Simonyan & Zisserman, 2014).

Despite the tremendous popularity of the above-mentioned approach, potentially negative transfer is often overlooked. It is not always well understood to which extent the retained knowledge about the source domain might negatively impact the target task. In particular, when the target dataset is limited, we expect the retrained knowledge to be significant. Negative transfer can hence take place when such knowledge interferes with the model's ability to adequately learn the target domain.

Various studies have focused on the effectiveness of TL (Huh et al., 2016; Kornblith et al., 2019; He et al., 2019; Yamada & Otani, 2022), aiming to shed light into its benefits and limitations. Notably, Kornblith et al. (2019) found that the pre-trained features are less general than previously suggested. A number of studies have focused specifically on the issue of negative transfer (Chen et al., 2019; Ganin et al., 2016; Liu et al., 2017; Zoph et al., 2020) and on exploring avenues to mitigate it.

We provide new means to analyze negative transfer in TL, by focusing on potential interference between the source and target domains. Our analysis demonstrates how this interference can be quantified and subsequently uncovers inherent weaknesses of models trained under the prevalent TL paradigm. This helps us predict which inputs are likely to be impacted by negative transfer and surface subtle failure cases of these models.

2 METHODOLOGY AND INSIGHTS

We design three experiments to shed light into domain interference in TL. These experiments are summarized in Figure 1. We demonstrate our approach on image classification with two architectures, ResNet-18 and VGG-16. We use ImageNet classification (Deng et al., 2009) as the source domain and classification of the Hymenoptera dataset (Chilamkurthy) as the target domain. This dataset contains 240 images in the training set, distributed equally between its two classes: ants and bees. We denote by $C_i^S : 1 \leq i \leq 1000$ and by $C_j^T : j \in \{\mathtt{a}, \mathtt{b}\}$ the categories of the source and target domains (ants and bees) respectively.

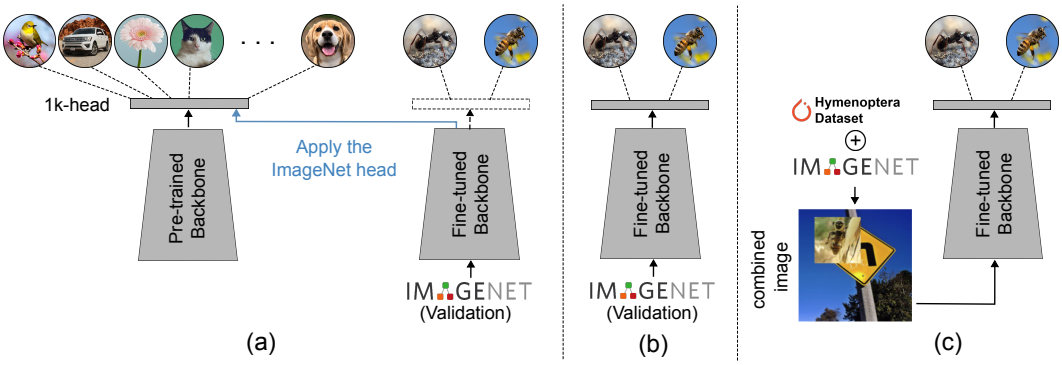

Figure 1: Illustrating our approach on a TL example from ImageNet to an ants-vs-bees classifier. (a) To expose the retained knowledge in the fine-tuned model, we replace its 2-way head with the ImageNet head, and use it to classify the source validation set. (b) To explore possible interference between the source and target domains, we feed the fine-tuned model with images from the source domain, keeping its 2-way head. (c) To measure such interference, we feed the fine-tuned model with images that contain the target categories in addition to potentially-confusing source categories.

We utilize four sets of models in our analysis:

- **ImageNet classifiers**: We use two ImageNet classifiers one based on ResNet-18 (RN-18 for short) and VGG-16, available as pretrained models in PyTorch (Paszke et al., 2019).
- **SimCLR backbones**: For further experimentation, we also use a ResNet-18 backbone pre-trained under SimCLR (Chen et al., 2020) in a self-supervised manner.
- **Fine-tuned models**: We fine-tune the above-mentioned three pre-trained models on the Hymenoptera dataset. The following table reports the accuracy of the fine-tuned models on the validation set. Appendix A.2 provides the training parameters used for fine-tuning.

Model	RN-18 (ImageNet)	VGG-16 (ImageNet)	RN-18 (SimClr)	Baseline
Accuracy	96%	95%	88%	79%

- **Baseline classifier**: This model serves as a baseline that is not exposed to ImageNet. Due to the limited training set, we train a small convolutional network to classify the Hymenoptera dataset from scratch (refer to Appendix A.1 for details on the architecture).

2.1 Exposing the retained knowledge

We aim to assess the source-domain knowledge retained by the pre-trained models after finetuning. For this purpose, we follow the process illustrated in figure 1a where we simply use the ImageNet head with the fine-tuned backbone instead of the pre-trained backbone. By feeding the validation set of ImageNet, we can calculate the recall of each ImageNet category. A high recall indicates that the category is preserved after fine-tuning. Table 1 summarizes the number of ImageNet classes having different ranges of recall. Evidently, the fine-tuned models are able to predict about half (resp. third) of the source-domain classes at a recall $\geq 50\%$. This indicates the strong retention of memories about the source domain.

Recall (%)	≤ 50	[50, 60[	[60, 70[	[70, 80[	[80, 90[	[90, 100]	All
RN-18 (ImageNet)	495	185	164	100	49	7	1000
VGG-16 (ImageNet)	671	128	97	63	33	7	1000

Table 1: A breakdown of ImageNet classes by their recall with the fine-tuned models.

2.2 ANALYZING DOMAIN INTERFERENCE

We feed the fine-tuned model with images from the ImageNet validation set, instead of ones from the Hymenoptera dataset (Figure 1b). Our hypothesis is that in the absence of domain interference, the model predictions for a source category C_i^S will be random, given that C_i^S is not semantically related with any target category C_j^T. For example, the "apiary (bee house)" and the "bee eater" ImageNet categories are semantically related with bees, with their image samples often including bees. In contrast, "street sign" and "traffic light" are semantically unrelated with bees.

In the presence of domain interference, the fine-tuned model tends to predict a specific target category C_j^T, often with high confidence, when fed with instances of a semantically unrelated category C_i^S. In that case we consider that C_i^S interferes with the predictions of the fine-tuned model, and can potentially be problematic when present in the input images, causing it to erroneously favor C_j^T.

To quantify domain interference in our models, we compute the bias $B_a^i = 1 - B_b^i$ of each ImageNet category C_i^S towards either one of the two target categories C_a^T (ants) and C_b^T (bees). For this purpose, we feed with the validation set instances X_i^{val} of this category to the fine-tuned model M and compute the bias based on its predictions as follows:

$$B_j^i = \frac{\left|x \in X_i^{val} : M(x) = C_j^T\right|}{|X_i^{val}|} \tag{1}$$

Table 2 shows for each of the four models we studied, a breakdown of ImageNet classes into five groups, depending on how highly they are biased toward either target class. It is evident that the two models that were pretrained on ImageNet exhibit significantly higher interference between its categories and the target ones, compared with the SimCLR model and our baseline model. By examining the interfering classes C_i^S in the former two models that exhibit high bias ($B_a^i > 0.8$ or $B_b^i > 0.8$) we found that the majority of them were semantically unrelated with the target classes. This indicates that the prediction of these two models might indeed be influenced by knowledge retained about various unrelated ImageNet classes (see Appendix A.3 for further analysis).

B_a^i (%)	0 to 20	20 to 40	40 to 60	60 to 80	80 to 100
Model (pertaining)	High bees bias	Slight bees bias	Neutral	Slight ants bias	High ants bias
RN-18 (ImageNet)	59	176	247	290	228
VGG-16 (ImageNet)	67	152	339	310	132
RN-18 (SimCLR)	3	118	503	355	21
Baseline (scratch)	13	103	628	253	3

Table 2: A breakdown of ImageNet classes by their bias toward the target classes with our classifiers. We highlight in blue values where pretraining on ImageNet incurred significant bias.

2.3 CONFUSING THE FINE-TUNED MODEL

We demonstrate how domain interference can confuse the fine-tuned models, leading them to erroneously favor specific target categories even when they are not present in the input. For this purpose, we feed the classifiers with images that contain the target categories superimposed as visual stimuli over ImageNet images as illustrated in Figure 1c. Our hypothesis is that the decision of the baseline model trained from scratch will not be strongly influenced by the background object, unlike the ones fine-tuned from ImageNet.

Creating Combined Images We select 20 images $\{Y_j^k : 1 \leq k \leq 20\}$ of each target category $C_j^T : j \in a, b$ that are recognizable by the classifiers with high scores. To test the detectability of C_j^T in the presence of an ImageNet category C_i^S, we superimpose each image Y_j^k at a random location on top of each instance of C_i^S in the ImageNet validation set, covering about 10% of the image area. We repeat the process 5 times, creating a set of $20 \times 50 \times 5 = 5000$ combined images $I_{(i,j)}$ having C_j^T as the ground-truth label. This set aims to test the source-target interference (C_i^S, C_j^T).

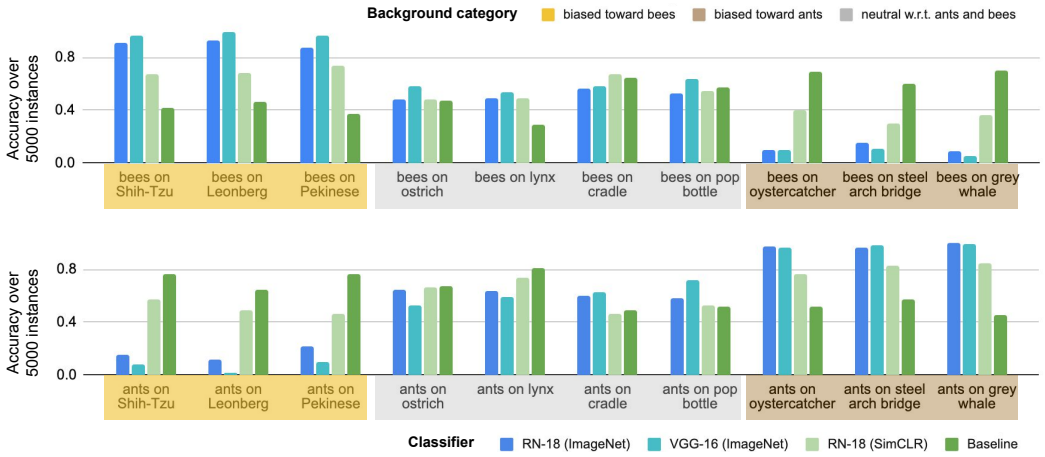

Figure 2: Comparing the accuracy of our four ants-vs-bees classifiers on different sets of inputs, curated to surface potential vulnerabilities of the two models pre-trained on ImageNet.

Figure 2 summarizes the accuracy of our models on various combined-image sets. We choose ten ImageNet categories to use as background objects (see Appendix A.4). Three of these categories demonstrate high bees bias ($B_b^i \geq 0.8$) with the two models pretrained on ImageNet, namely "Shih-Tzu", "Leonberg", and "Pekinese". Likewise, three of the categories demonstrate high ants bias ($B_a^i \geq 0.8$) with both models, namely "oystercatcher", "steel arch bridge", and "grey whale". The remaining four categories are neutral ($40 \leq B_a^i < 60$ and $40 \leq B_b^i < 60$). Predictably, the accuracy of the models on superimposed images is lower than on the Hymenoptera validation set.

It is evident that the two models pre-trained on ImageNet have much lower accuracy than the other two models on image sets $I_{(i,j)}$ when the background category C_i^S is highly biased against the superimposed target category C_j^T. In contrast, all four models have relatively comparable accuracy when the background category is neutral. Remarkably, the ResNet-18 model fine-tuned from SimCLR is less sensitive to the presence of background objects than the two models pre-trained on ImageNet, suggesting that self-supervised learning can mitigate domain interference.

Takeway Analyzing domain interference helps us uncover potential negative transfer in models trained under supervised TL. Besides synthetic test cases, we can predict actual failure cases in real inputs. Figure 3 demonstrates how the models fine-tuned from ImageNet are mostly unable to detect bees on `street sign` in real images, as well as bees with `oystercatcher` in generated images, both categories incur high ants bias ($B_a^i \geq 0.8$) within these models.

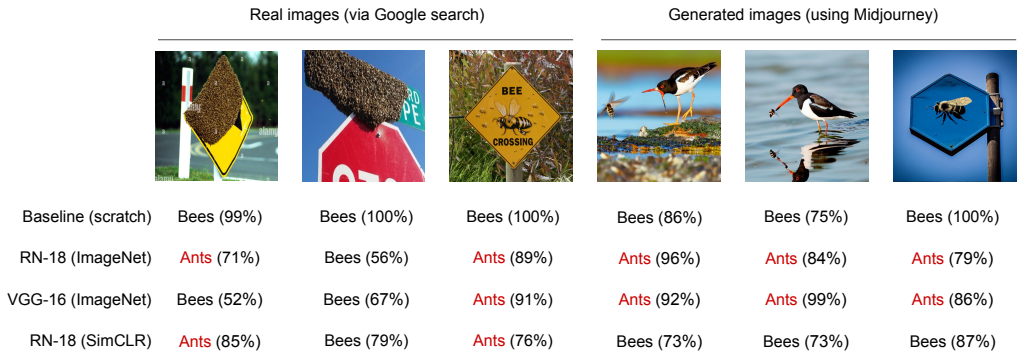

Figure 3: Model prediction on example images that contain bees along with ant-biased objects (`street sign` and `oystercatcher`). While the baseline model is able to classify these examples as `bees`, the two models fine-tuned from ImageNet were frequently confused.

REFERENCES

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Xinyang Chen, Sinan Wang, Bo Fu, Mingsheng Long, and Jianmin Wang. Catastrophic forgetting meets negative transfer: Batch spectral shrinkage for safe transfer learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Sasank Chilamkurthy. Transfer learning for computer vision tutorial. URL `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html`. PyTorch Tutorials.

N Narayan Das, Naresh Kumar, Manjit Kaur, Vijay Kumar, and Dilbag Singh. Automated deep transfer learning-based approach for detection of covid-19 infection in chest x-rays. *Innovation and Research in BioMedical engineering*, 2020.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255. IEEE, 2009.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030, 2016.

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*, pp. 770–778, 2016.

Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4918–4927, 2019.

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

Aayush Jaiswal, Neha Gianchandani, Dilbag Singh, Vijay Kumar, and Manjit Kaur. Classification of th†e covid-19 infected patients using densenet201 based deep transfer learning. *Journal of Biomolecular Structure and Dynamics*, 39(15):5682–5689, 2021.

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*, pp. 2661–2671, 2019.

Jiaming Liu, Yali Wang, and Yu Qiao. Sparse deep transfer learning for convolutional neural network. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, et al. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8024–8035, 2019.

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

Uppala Sumanth, Narinder Singh Punn, Sanjay Kumar Sonbhadra, and Sonali Agarwal. Enhanced behavioral cloning-based self-driving car using transfer learning. In *Data Management, Analytics and Innovation*, pp. 185–198. Springer, 2022.

Yutaro Yamada and Mayu Otani. Does robustness on imagenet transfer to downstream tasks? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9215–9224, 2022.

Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. Rethinking pre-training and self-training. *Advances in neural information processing systems*, 33: 3833–3845, 2020.

A Appendix

A.1 Baseline Hymenoptera Classifier

Our baseline network consists of three convolutional layers with strides equal to 1, and 32, 64, and 128 filters respectively, all the filters are of size 3×3. Each convolutional layer is followed by a max pooling layer of size 2×2. The final two layers are dense layers with 512, and 128 neurons respectively, and 0.5 dropout for each of them. The last layer performs a softmax function with two categories (i.e. `ants` and `bees`). The model was trained using the configurations shown in table 3 and achieved 79% accuracy on the validation set.

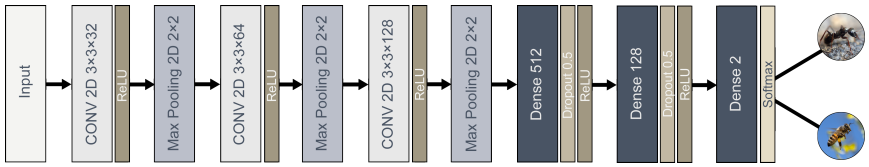

Figure 4: The architecture of the baseline model we trained from scratch on the target dataset.

A.2 Training Hyperparameters

				LR decay		
Model (pretraining)	Optimizer	LR	momentum	period	gamma	epochs
RN-18 (ImageNet)	SGD	0.001	0.9	7	0.1	25
VGG-16 (ImageNet)	SGD	0.001	0.9	7	0.1	25
RN-18 (SimCLR)	RMSProp	$1e-4$	-	-	-	50
Baseline (scratch)	RMSProp	$1e-4$	-	-	-	50

Table 3: The configurations used to train our models.

A.3 Comparing Bias Across Architectures

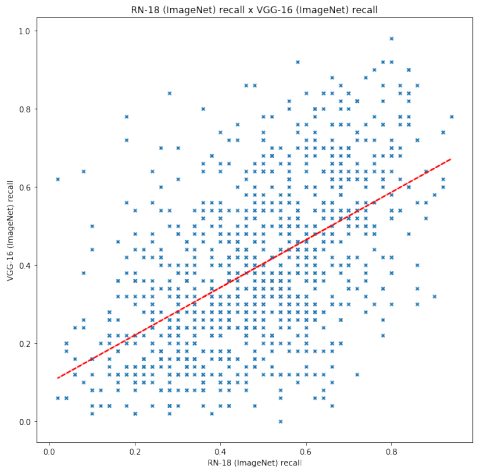

Figure 5: A scatter plot of ImageNet categories showing their recall (Section 2.1) in two models, ResNet-18 vs. VGG-16. Both models are pretrained on ImageNet. The red line indicates a positive correlation ($R = 0.56$), which suggests that the knowledge retained is, in part, architecture-agnostic.

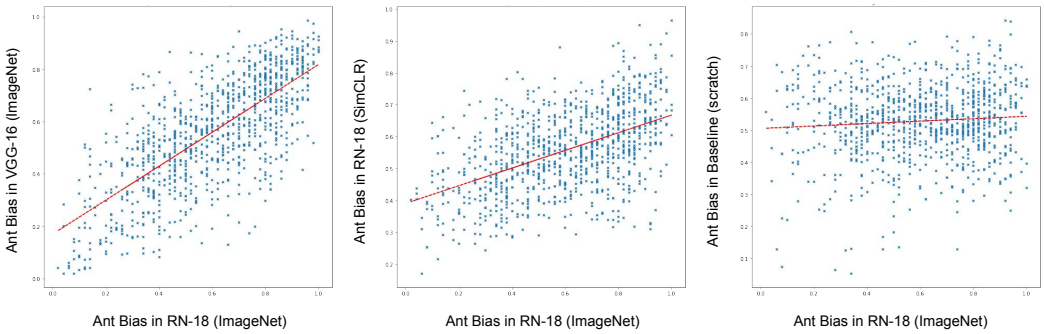

Figure 6: Scatter plots of ImageNet categories comparing their ant bias (Section 2.2) in ResNet-18 fine-tuned from ImageNet with the three other models in our study. Interestingly, this bias is highly correlated between the two models pretrained on ImageNet ($R = 0.72$), which suggests that it does not stem from the architecture, but is rather inherent to TL from ImageNet. Also remarkably, the bias is not correlated between ResNet-18 (ImageNet) and the baseline model ($R = 0.08$). Finally, the bias when finetuning from SimCLR is moderately correlated with the bias when finetuning from ImageNet ($R = 0.5$).

A.4 Ant Bias for Selected Categories

ImageNet Category	RN-18 (ImageNet)	VGG-16 (ImageNet)	RN-18 (SimCLR)	Baseline
oystercatcher	1	0.87	0.68	0.4
grey whale	1	0.92	0.7	0.32
steel arch bridge	0.98	0.82	0.74	0.41
Shih-Tzu	0.02	0.04	0.4	0.56
Leonberg	0.04	0.02	0.42	0.53
Pekinese	0.06	0.04	0.31	0.62
Ostrich	0.48	0.45	0.58	0.57
pop bottle	0.52	0.54	0.5	0.42
lynx	0.48	0.45	0.56	0.67
cradle	0.52	0.49	0.39	0.4

Table 4: The ant bias B_a^i of the ImageNet categories used in Section 2.3. Values colored in brown indicate high ant bias. Values colored in yellow indicate low ant bias, and hence high bee bias.