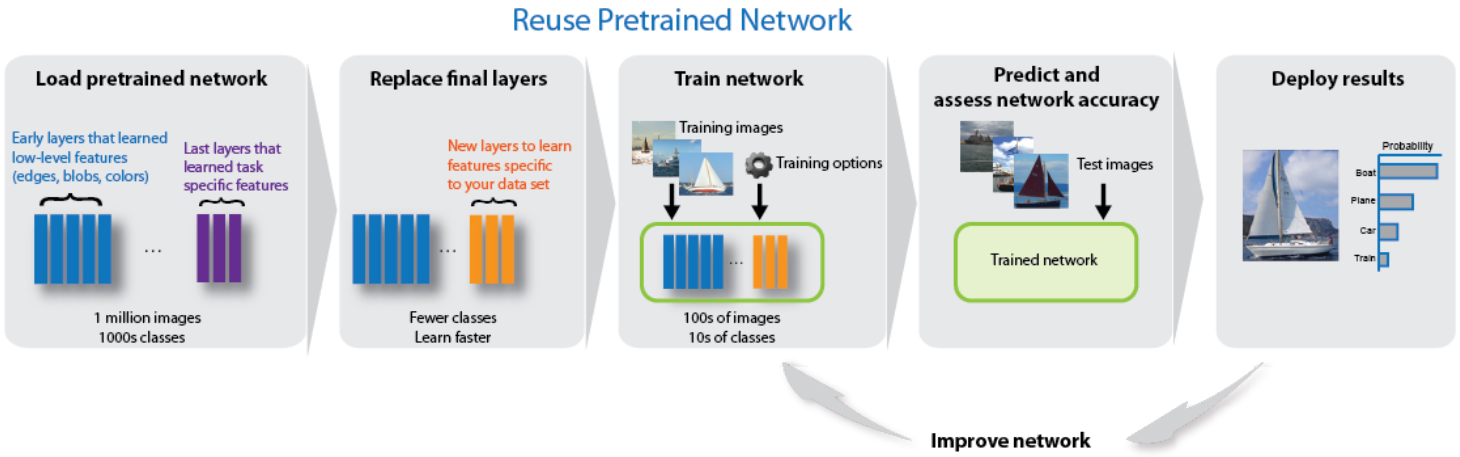
S15

Submit Assignment

Due Sunday by 11:59pm **Points** 10,000 **Submitting** a website url
Available May 3 at 10am - May 17 at 11:59pm 15 days

TRANSFER LEARNING



Disclaimer: Not falling into NIH syndrome, we would majorly refer to [this (https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a)](https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a) excellent source for TF content.

One of the most powerful ideas in deep learning is that sometimes we can the knowledge
of the neuron network has learned from one task and apply that knowledge to a separate task.
This is called Transfer Learning

We know that humans don't learning everything from the ground up, and leverage and transfer their knowledge from previously learnt domain to newer domains and tasks.
Transfer learning, it is believed, can further our progress towards AGI.

In fact, transfer learning is not a concept which just cropped up in the 2010s. The Neural Information
Processing Systems (NIPS) 1995 workshop *Learning to Learn: Knowledge Consolidation and Transfer in*
Inductive Systems is believed to have provided the initial motivation for research in this field.
Since then, terms such as *Learning to Learn*, *Knowledge Consolidation*,

and *Inductive Transfer*
have been used interchangeably with transfer learning. Invariably, different researchers
and academic texts provide definitions from different contexts.

In their famous book, *Deep Learning*, Goodfellow et al refer to transfer learning in the context of generalization.

Their definition is as follows:

> Situation where what has been learned in one setting is exploited to improve generalization in another setting.

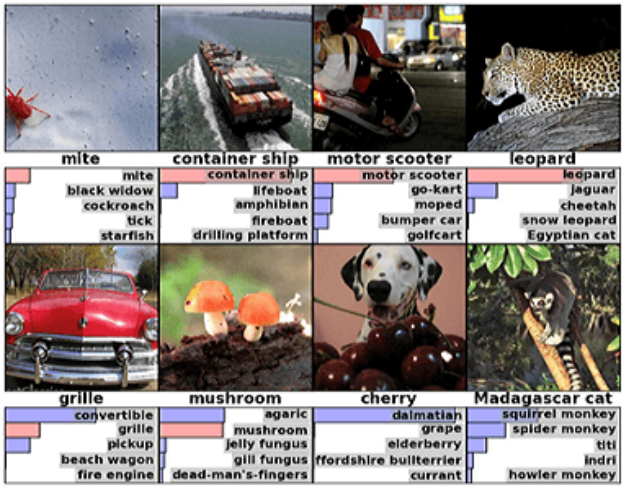
Thus, the key motivation, especially considering the context of deep learning is the fact that most models which solve complex problems need a whole lot of data, and getting vast amounts of labeled data for supervised models can be really difficult, considering the time and effort it takes to label data points.

A simple example would be the ***ImageNet dataset*** **(http://image-net.org/index)** **,** which has millions of images
pertaining to different categories, thanks to years of hard work starting at Stanford!

ImageNet Challenge

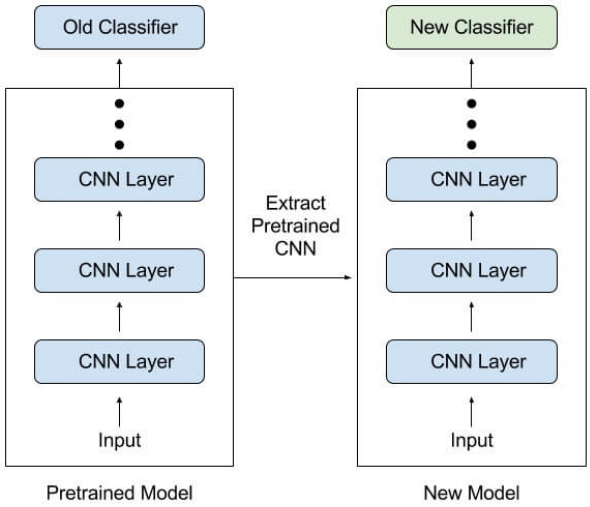
IMAGENET

- 1,000 object classes (categories).
- Images:
  - 1.2 M train
  - 100k test.



However, getting such a dataset for every domain is tough. Besides, most deep learning
models are very specialized to a particular domain or even a specific task.

While these might be state-of-the-art models, with really high accuracy and beating all benchmarks
, it would be only on very specific datasets and end up suffering a significant loss in performance
when used in a new task which might still be similar to the one it was trained on.



Traditional Learning is isolated, and occurs purely based on specific tasks, datasets and training separate isolated models on them. No knowledge is retained which can be transferred from one model to another.

In **transfer learning**, you can leverage knowledge (features, weights, etc) from previously trained models for training newer models and even tackle problems like having less data for the newer task.

Formal Definition

Given a source domain $\mathcal{D}_S$, a corresponding source task $\mathcal{T}_S$, as well as a target domain $\mathcal{D}_T$ and a target task $\mathcal{T}_T$, the objective of transfer learning now is to enable us to learn the target conditional probability distribution $P(Y_T|X_T)$ in $\mathcal{D}_T$ with the information gained from $\mathcal{D}_S$ and $\mathcal{T}_S$ where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$. In most cases, a limited number of labeled target examples, which is exponentially smaller than the number of labeled source examples are assumed to be available.

There are 3 main questions we need to answer in TL

1. **What to transfer**: we need to identify the portion of knowledge is source-specific
   and what is common between the source and the target. This helps us decide
   which layers we can keep, which we need to retrain and which we need to completely remove
2. **When to transfer**: Transferring for the sake of it my make matters worse than
   improving anything (negative transfer). We should have a reason for transfer.
   1. lack of dataset

2. lack of time
3. lack of training hardware
4. difficult in retraining newer model from scratch, etc

3. **How to transfer**: These are the changes which are required in the existing algorithms

Transfer Learning Strategies

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

TABLE 2
Different Settings of Transfer Learning

Transfer Learning Settings	Related Areas	Source Domain Labels	Target Domain Labels	Tasks
Inductive Transfer Learning	Multi-task Learning	Available	Available	Regression, Classification
	Self-taught Learning	Unavailable	Available	Regression, Classification
Transductive Transfer Learning	Domain Adaptation, Sample Selection Bias, Co-variate Shift	Available	Unavailable	Regression, Classification
Unsupervised Transfer Learning		Unavailable	Unavailable	Clustering, Dimensionality Reduction

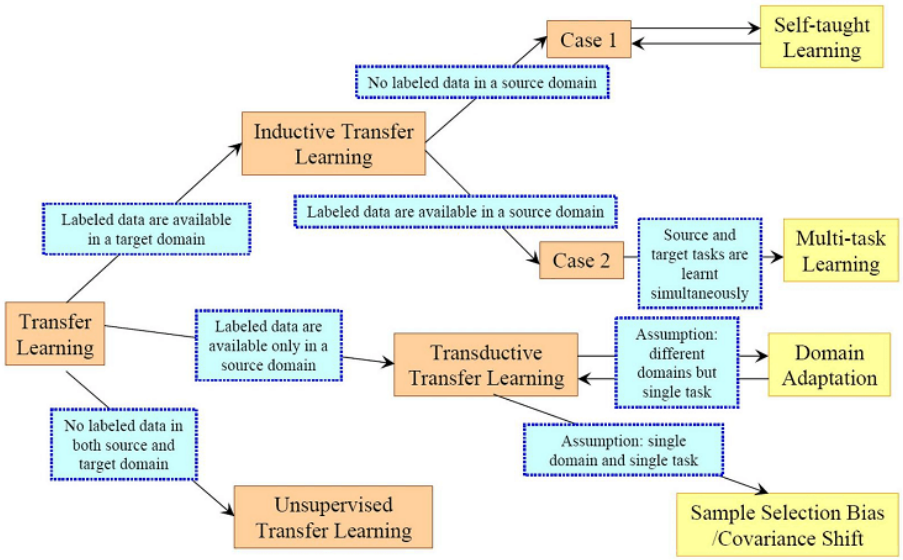


Fig. 2. An Overview of Different Settings of Transfer

TABLE 3
Different Approaches to Transfer Learning

Transfer Learning Approaches	Brief Description
Instance-transfer	To re-weight some labeled data in the source domain for use in the target domain [6], [28], [29], [30], [31], [24], [32], [33], [34], [35].
Feature-representation-transfer	Find a "good" feature representation that reduces difference between the source and the target domains and the error of classification and regression models [22], [36], [37], [38], [39], [8], [40], [41], [42], [43], [44].
Parameter-transfer	Discover shared parameters or priors between the source domain and target domain models, which can benefit for transfer learning [45], [46], [47], [48], [49].
Relational-knowledge-transfer	Build mapping of relational knowledge between the source domain and the target domains. Both domains are relational domains and i.i.d assumption is relaxed in each domain [50], [51], [52].

TABLE 4
Different Approaches Used in Different Settings

	Inductive Transfer Learning	Transductive Transfer Learning	Unsupervised Transfer Learning
Instance-transfer	√	√	
Feature-representation-transfer	√	√	√
Parameter-transfer	√		
Relational-knowledge-transfer	√		

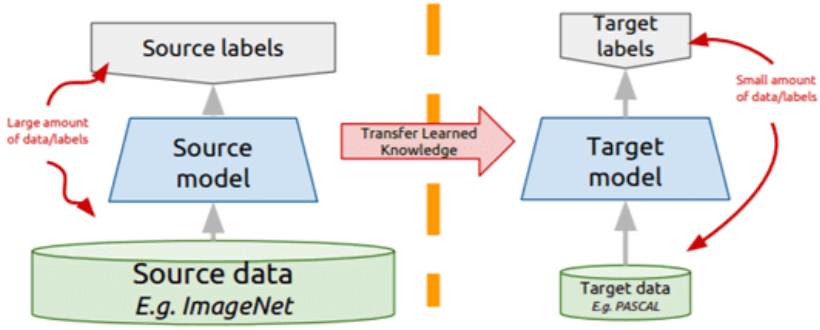
Transfer learning: idea

Instead of training a deep network from scratch for your task:

- Take a network trained on a different domain for a different **source task**
- Adapt it for your domain and your **target task**

Variations:

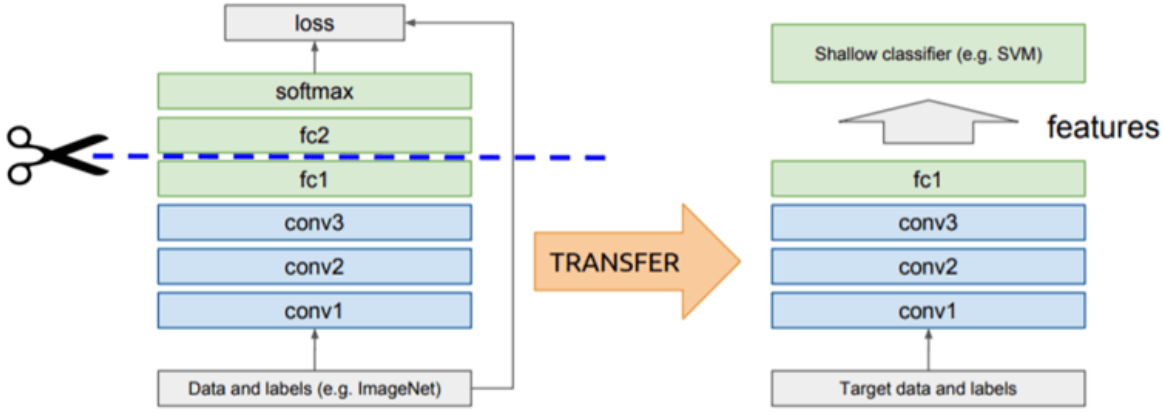
- Same domain, different task
- Different domain, same task



Off-the-shelf Pre-trained Models as Feature Extractors

Idea: use outputs of one or more layers of a network trained on a different task as generic feature detectors. Train a new shallow model on these features.

Assumes that $D_S = D_T$

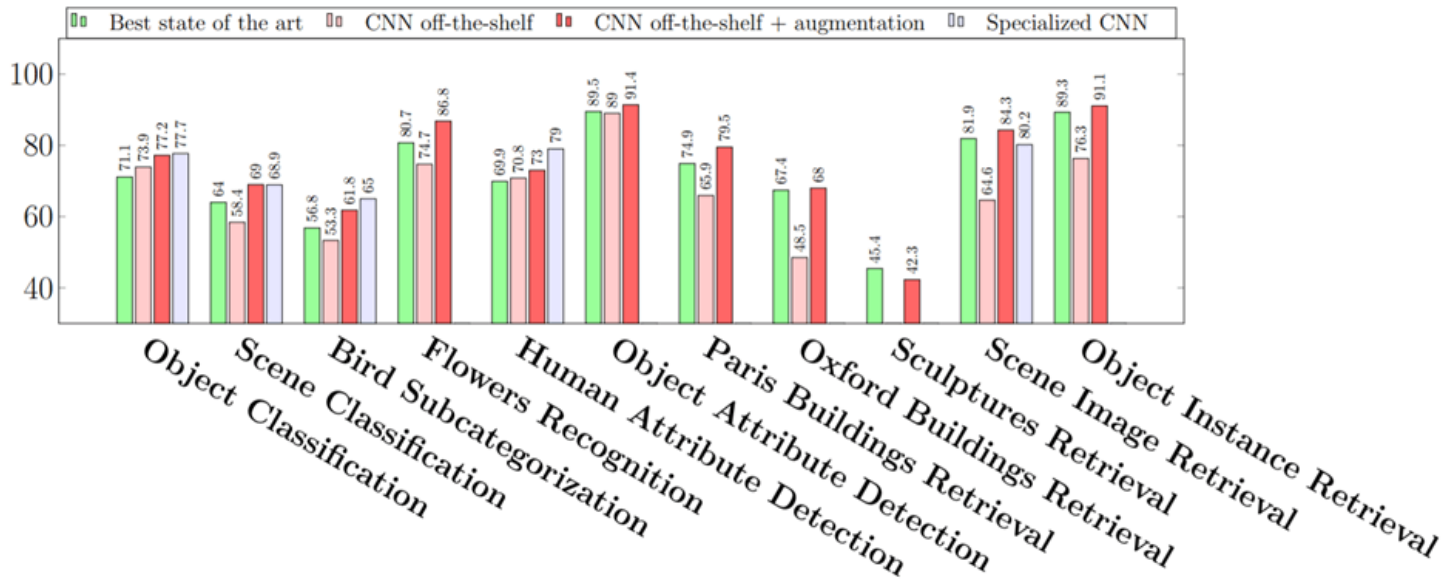


> The key idea here is to just leverage the pre-trained model's weighted layers to extract features but not to update the weights of the model's layers during training with new data for the new task.

Now a question might arise,

how well do these pre-trained off-the-shelf features really work

in practice with different tasks?



Based on the red and pink bars in the above figure, you can clearly see that the features from the pre-trained models consistently out-perform very specialized task-focused deep learning models.

Fine Tuning Off-the-shelf Pre-trained Models

This is a more involved technique, where we do not just replace the final layer (for classification/regression), but we also selectively retrain some of the previous layers. Deep neural networks are highly configurable architectures

with various hyper parameters.

Fine-tuning: supervised domain adaptation

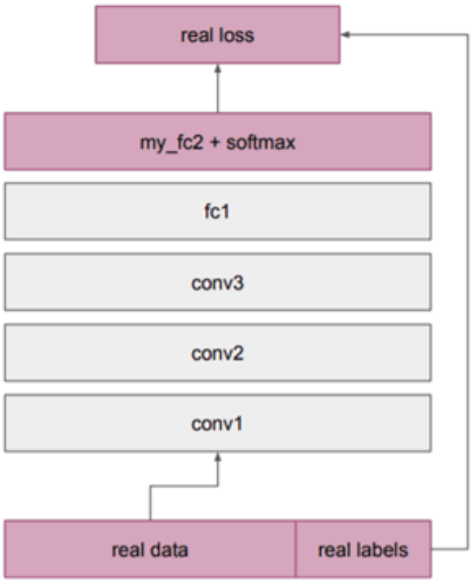
Train deep net on "nearby" task for which it is easy to get labels using standard backprop

- E.g. ImageNet classification
- Pseudo classes from augmented data
- Slow feature learning, ego-motion

Cut off top layer(s) of network and replace with supervised objective for target domain

Fine-tune network using backprop with labels for target domain until validation loss starts to increase

Aligns D_S with D_T



Freezing or Fine-tuning?

This brings us to the question, should we freeze layers in the network to use them as feature extractors or should we also fine-tune layers in the process?

Freeze or fine-tune?

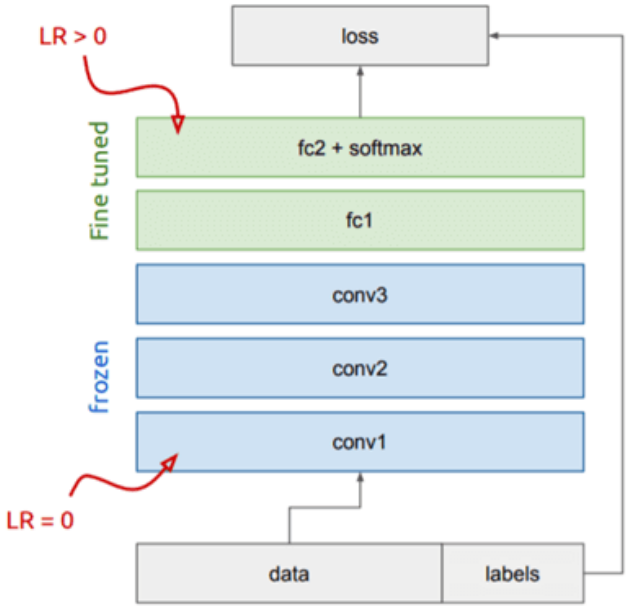
Bottom *n* layers can be frozen or fine tuned.

- **Frozen**: not updated during backprop
- **Fine-tuned**: updated during backprop

Which to do depends on target task:

- **Freeze**: target task labels are scarce, and we want to avoid overfitting
- **Fine-tune**: target task labels are more plentiful

In general, we can set learning rates to be different for each layer to find a tradeoff between freezing and fine tuning



Pre-trained Models

For computer vision, you can leverage some popular models including,

VGG-16 (https://www.kaggle.com/keras/vgg16/home)

VGG-19 (https://www.kaggle.com/keras/vgg19/home)

Inception V3 (https://arxiv.org/abs/1512.00567)

XCeption (https://arxiv.org/abs/1610.02357)

ResNet-50 (https://www.kaggle.com/keras/resnet50/home)

For natural language processing tasks, things become more difficult due to the varied

nature of NLP tasks. You can leverage word embedding models including,

Word2Vec (https://en.wikipedia.org/wiki/Word2vec)

GloVe (https://nlp.stanford.edu/projects/glove/)

FastText (https://fasttext.cc/)

Universal Sentence Encoder by Google (https://arxiv.org/abs/1803.11175)

Bidirectional Encoder Representations from Transformers (BERT) by Google (https://arxiv.org/abs/1810.04805)

Most of the resource above and later is taken from this excellent primer [**SOURCE** (https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a)].
Let check out the Case Study 1 & Case Study 2 there to understand all of this better.

Transfer Learning Advantages

We have already covered several advantages of transfer learning in some way or the other in the previous sections. Typically transfer learning enables us to build more robust models that can perform a wide variety of tasks.

- Helps solve complex real-world problems with several constraints
- Tackle problems like having little or almost no labeled data availability
- Ease of transferring knowledge from one model to another based on domains and tasks
- It provides a path towards achieving Artificial General Intelligence someday in the future!

ASSIGNMENT 15

- Assignment description is already shared in the last assignment:
  - Given an image with foreground objects and background image, predict the depth map as well as a mask for the foreground object.
- It is an open problem and you can solve it any way you want.
- Let's look at how it can be approached through some examples.
- Assignment 14 (15A)was given to start preparing you for assignment 15th. 14th (15A) automatically becomes critical to work on the 15th.
- **The 15th assignment is NOT a group assignment**. You are supposed to submit it along.
- What happens when you copy? Well, WHAT happens when we copy? WHO knows!
- This assignment is worth 10,000 points.
- Assignment 15th is THE qualifying assignment.

Evaluation:

- A very heavy score on documenting what you have done. If you submit just files without documenting what you have (gone through to do/) done, then even if you get perfect results, you won't get more than 40%.
- A heavy score for the use of Modular code (especially if it is from the package you have written)
- A heavy score for your data management skills
  - have you thought about changing data format, channels, etc
  - have you thought and utilized the fact that you can train on smaller images first and then move to large resolution ones
  - how creative you have been, for example, what all loss functions have you tried, have you tried to only solve for 1 problem before solving for both, etc
  - how creative your DNN is, how many params you have (a DNN without Param count mentioned will not get any evaluation points from us as we won't know did it actually help)
  - how creative you have been while using data augmentation, and why did you pick those augmentations
  - colab gives you less than 9-10 hours, then how did you manage creatively compute for days
  - have you done any analysis on how much time each block (dnn, data prep, loss calc, etc) takes (basic python "timeit" at least!)
  - how have you presented your results?
    - we are now talking about the accuracy of the depth map and area of the foreground. How would you present your accuracy now?
    - have you just thrown some logs/numbers on the screen or have actually presented the visual results as well.

Assignment 15 will test whether you have come out of the "novice" shell or not. Anything which looks too primitive or "not well thought off", will attract a negative penalty.

Once done submit your assignment 15 GitHub link. We will **ONLY** read your readme file, and unless your readme file directs us to read any other file/notebook we will not open anything.

Evaluators are lazy, to make sure that you can keep them excited when they are reading your readme file, and think how to force them to look at your code.

SESSION VIDEO: