# Transfer Learning

Peerapon Vateekul, Ph.D
Nvidia IVA workshop
(TAs: Amarin Jettakul, Itsara Wichakam)

CHULA ENGINEERING
Foundation toward Innovation
COMPUTER

# Outline

1. Introduction of Transfer Learning
2. How to choose backbone model
3. Same Task, Different domain
4. Different Task
5. Why adaptation is so important
6. Other types of transfer learning
7. Conclusion

# 1 Introduction

# Transfer learning

***Myth:*** you can't do deep learning unless you have a million labelled examples for your problem.

***Reality:***

- You can learn useful representations from **unlabelled data**
- *You can **transfer** learned representations from a related task*
- You can train on a nearby **surrogate objective** for which it is easy to generate labels

# Transfer learning: idea

Instead of training a deep network from scratch for your task, you can

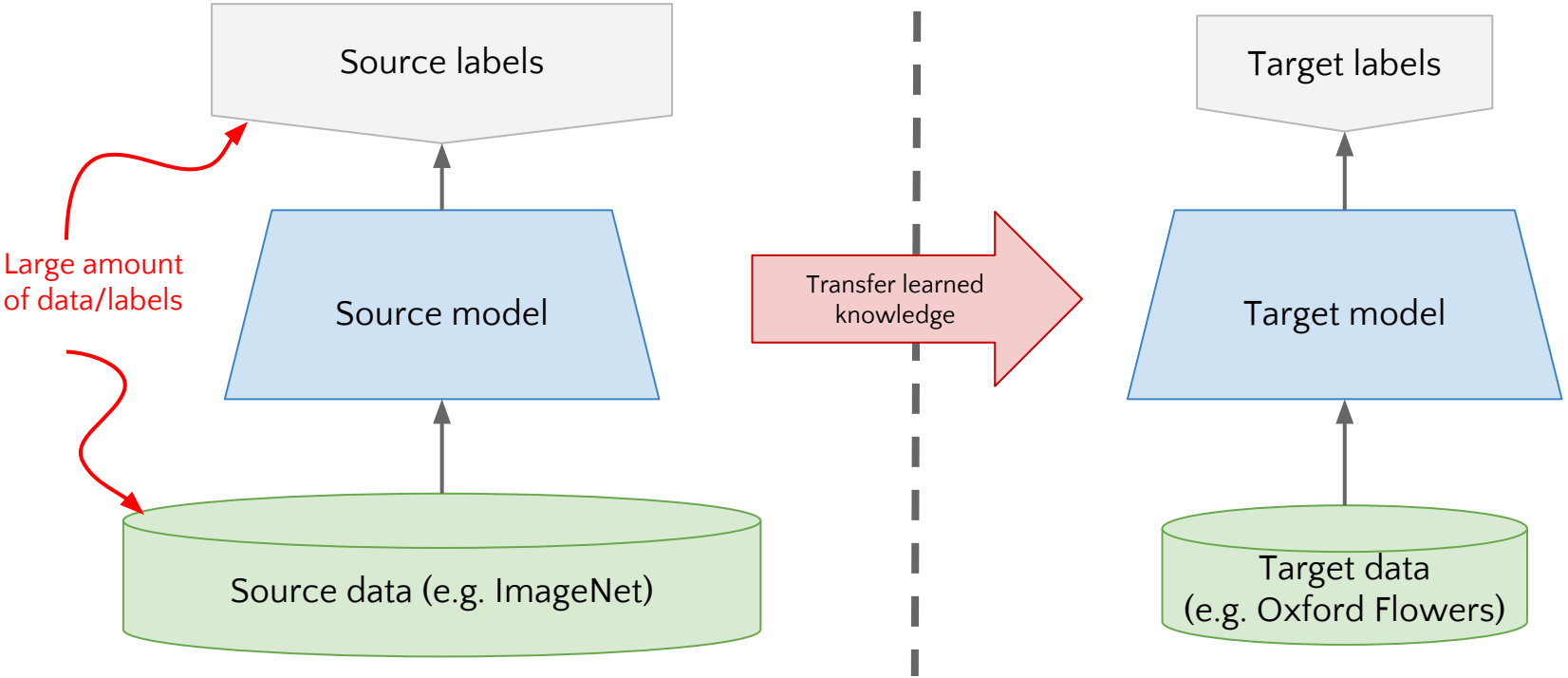
- Take a network trained on a different domain for a different **source task**
- Adapt it for your domain and your **target task**

This lecture will talk about how to do this.

Variations:

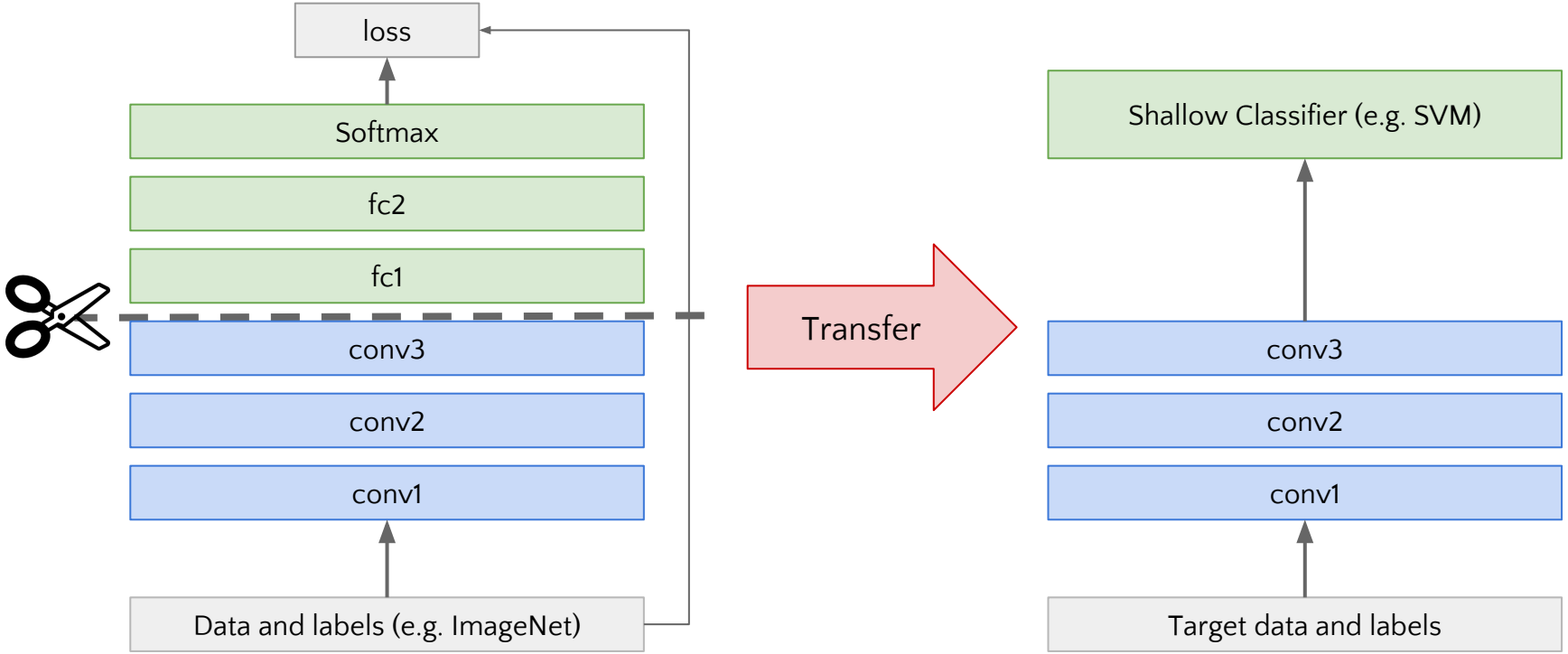
- Same domain, different task
- Different domain, same task

# Transfer learning: idea



# “Off-the-shelf (Feature Extractor)”

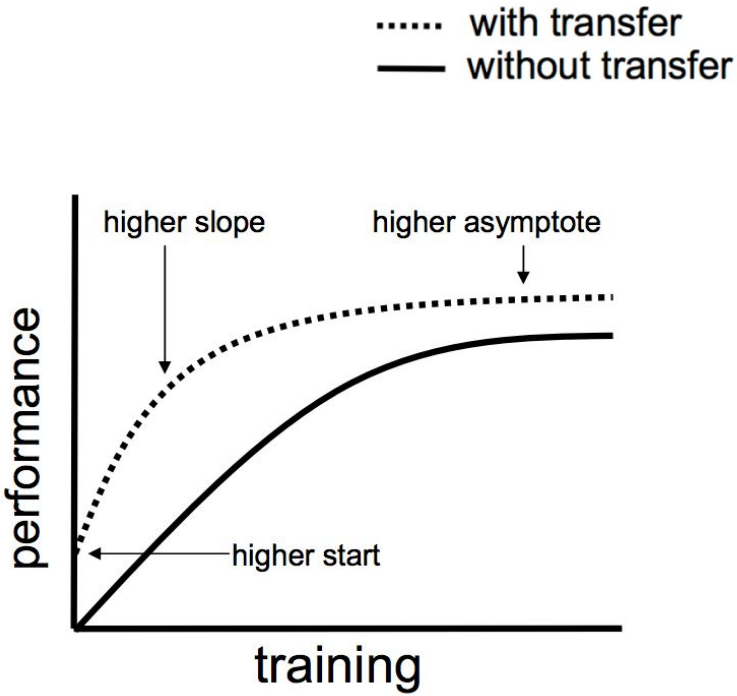
Idea: use outputs of one or more layers of a network trained on a different task as generic feature detectors. Train a new shallow model on these features.



# Transfer learning: 3 benefits

Lisa Torrey and Jude Shavlik in their chapter on transfer learning describe **three possible benefits** to look for when using transfer learning:

1. Higher start. The initial skill (before refining the model) on the source model is higher than it otherwise would be.
2. Higher slope. The rate of improvement of skill during training of the source model is steeper than it otherwise would be.
3. Higher asymptote. The converged skill of the trained model is better than it otherwise would be.




Chapter 11: Transfer Learning, Handbook of Research on Machine Learning Applications, 2009.

# Can we do better than off the shelf features ?

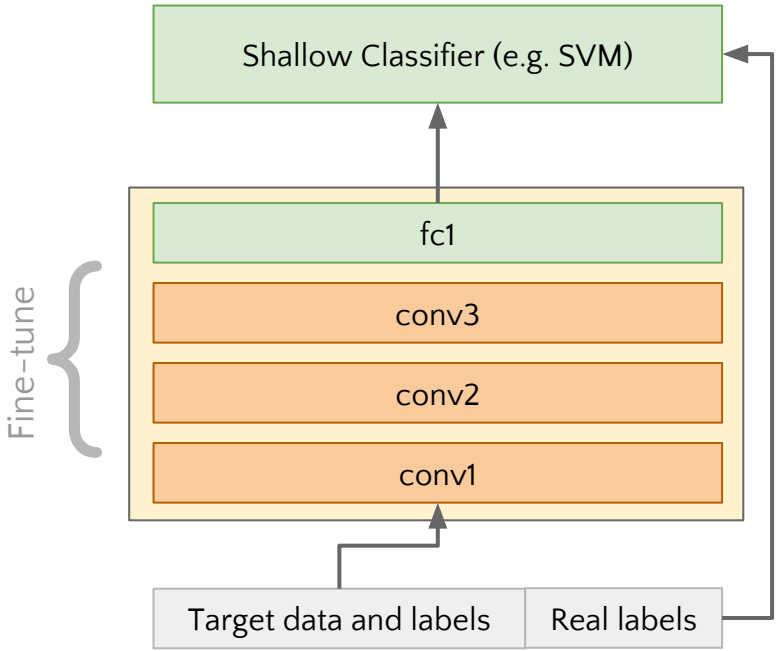
# Fine-tuning: supervised task adaptation

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.



# How transferable are features

**Lower layers: more general features.**

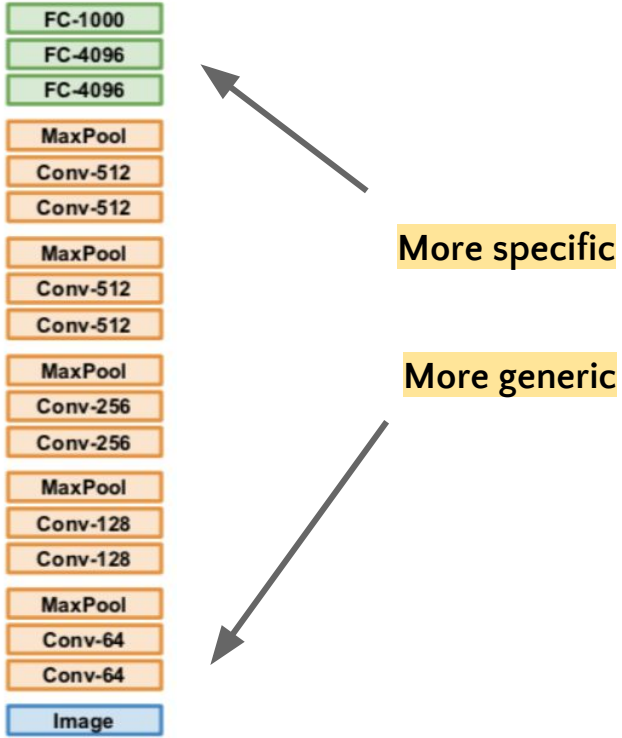
Transfer very well to other tasks.

**Higher layers: more task specific.**

Fine-tuning improves generalization when sufficient examples are available.

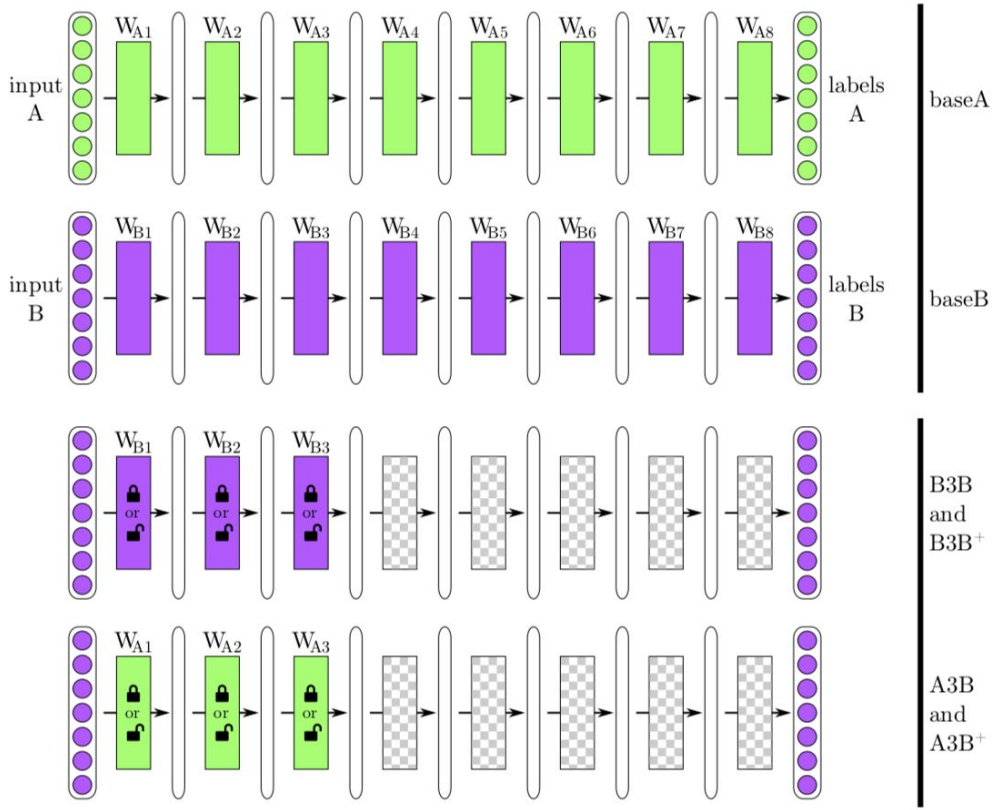
Transfer learning and fine tuning often lead to better performance than training from scratch on the target dataset.

Even features transferred from distant tasks are often better than random initial weights!




Yosinki et al., **How transferable are features in deep neural networks,** NIPS 2014 https://arxiv.org/abs/1411.1792

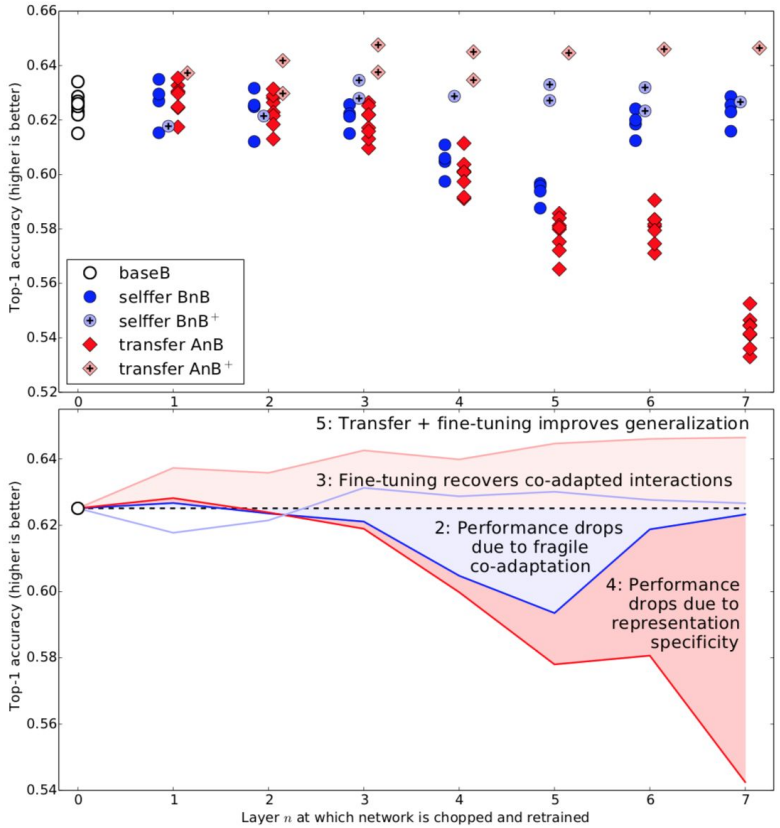
# How transferable are features




- **A *selffer* network B3B**: the first 3 layers are copied from baseB and frozen. The five higher layers (4–8) are initialized randomly and trained on dataset B. This network is a control for the next transfer network.
- **A *transfer* network A3B**: the first 3 layers are copied from base A and frozen. The five higher layers (4–8) are initialized randomly and trained toward dataset B. Intuitively, here we copy the first 3 layers from a network trained on dataset A and then learn higher layer features on top of them to classify a new target dataset B. If A3B performs as well as baseB, there is evidence that the third-layer features are general, at least with respect to B. If performance suffers, there is evidence that the third-layer features are specific to A.
- **B3B⁺ and A3B⁺** are just like B3B and A3B, but where all transferred layers are ***fine-tuned***.

Yosinki et al., **How transferable are features in deep neural networks,** NIPS 2014 https://arxiv.org/abs/1411.1792

# How transferable are features



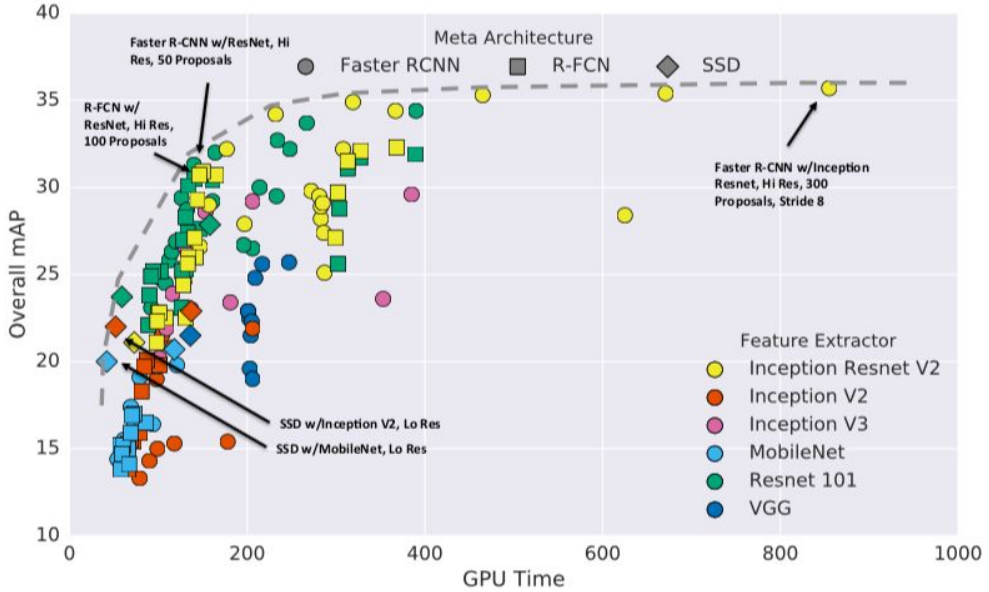

Yosinki et al., **How transferable are features in deep neural networks,** NIPS 2014 https://arxiv.org/abs/1411.1792

2

# How to choose backbone model

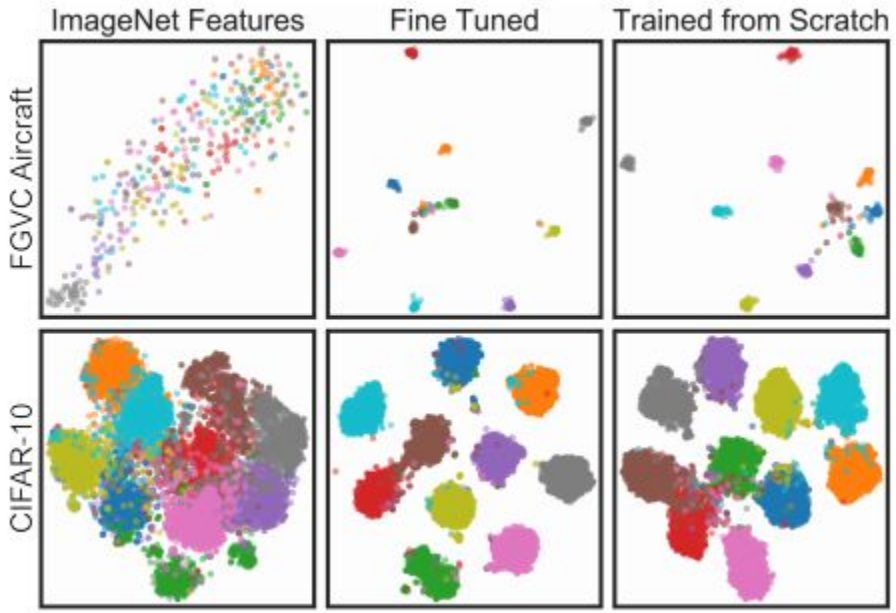
# How to choose backbone (pretrained weights) model

- Choose model to transfer based on for target application (e.g. smaller network in case you want to run on mobile, and bigger network is required if running on server)
- The graph below shows the backbone models performance measured on the COCO detection task.



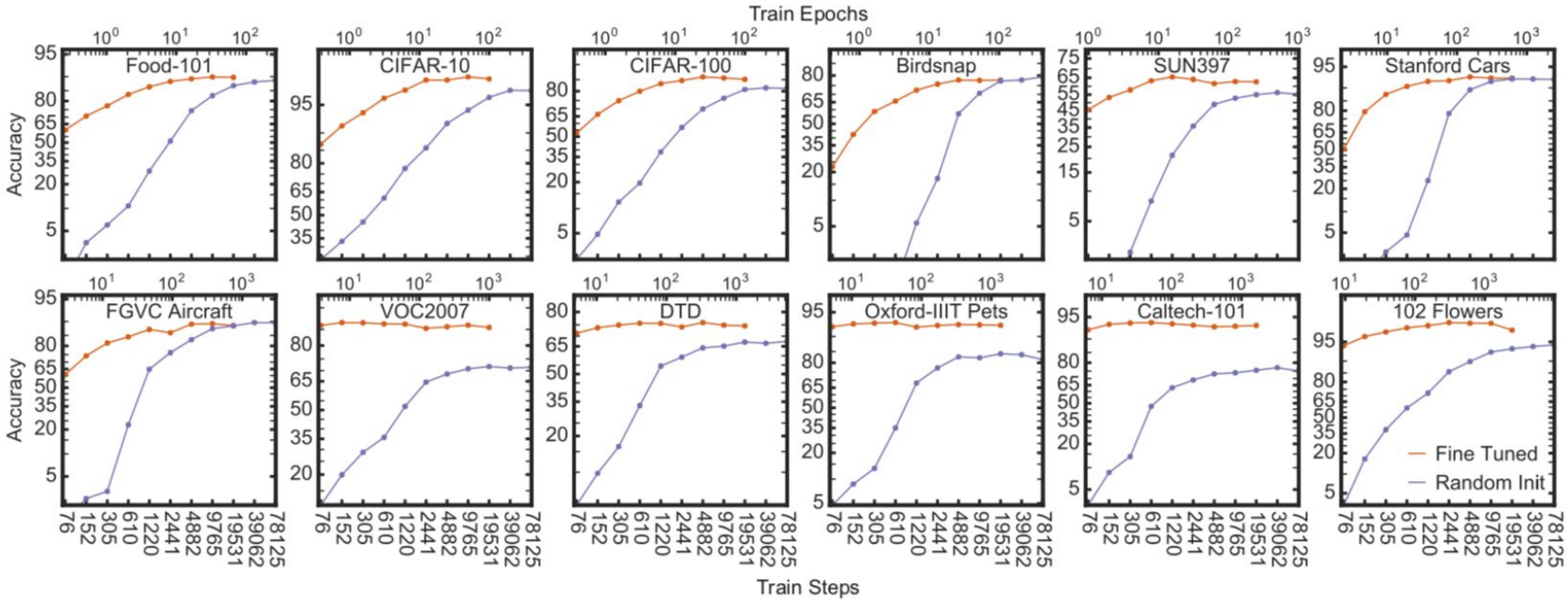

Huang et al., **Speed/accuracy trade-offs for modern convolutional object detectors**, CVPR 2017.

# Do Better ImageNet Models Transfer Better?



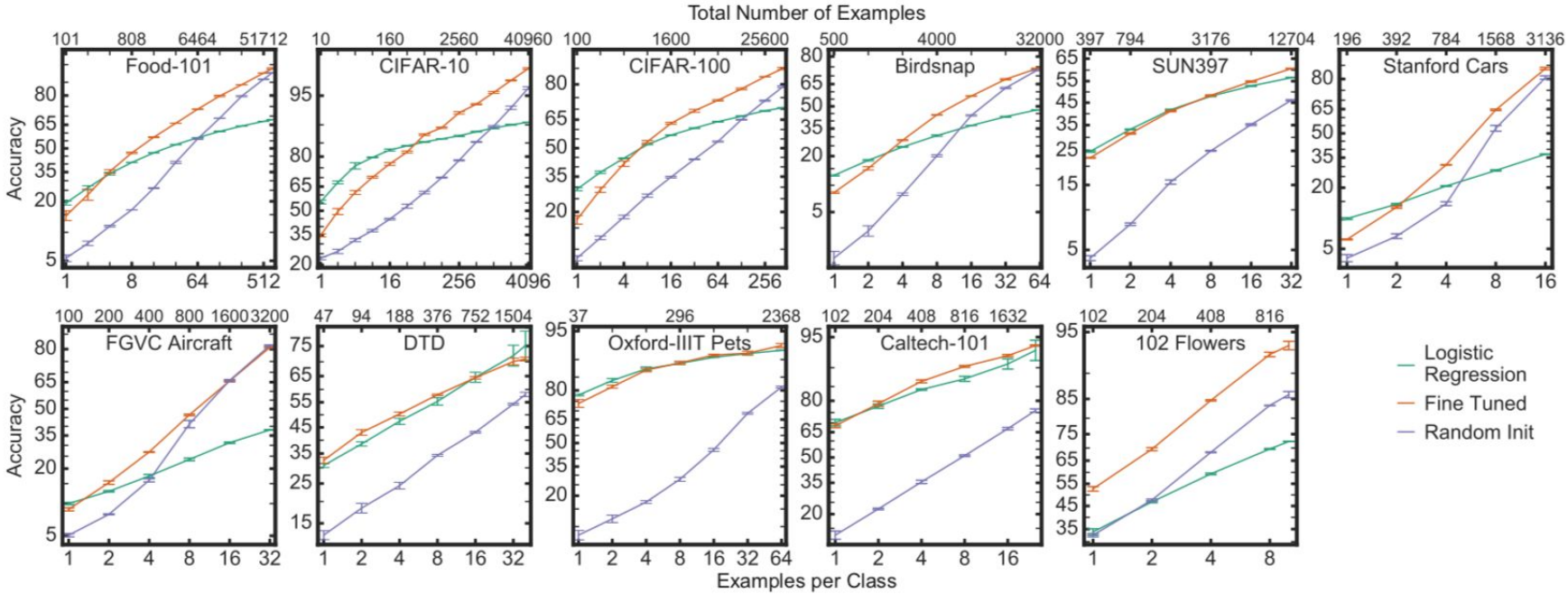

Kornblith et al., Do Better ImageNet Models Transfer Better?, arxiv 2018 https://arxiv.org/abs/1805.08974

# Do Better ImageNet Models Transfer Better? (cont.)




Kornblith et al., Do Better ImageNet Models Transfer Better?, arxiv 2018 https://arxiv.org/abs/1805.08974

# Do Better ImageNet Models Transfer Better? (cont.)



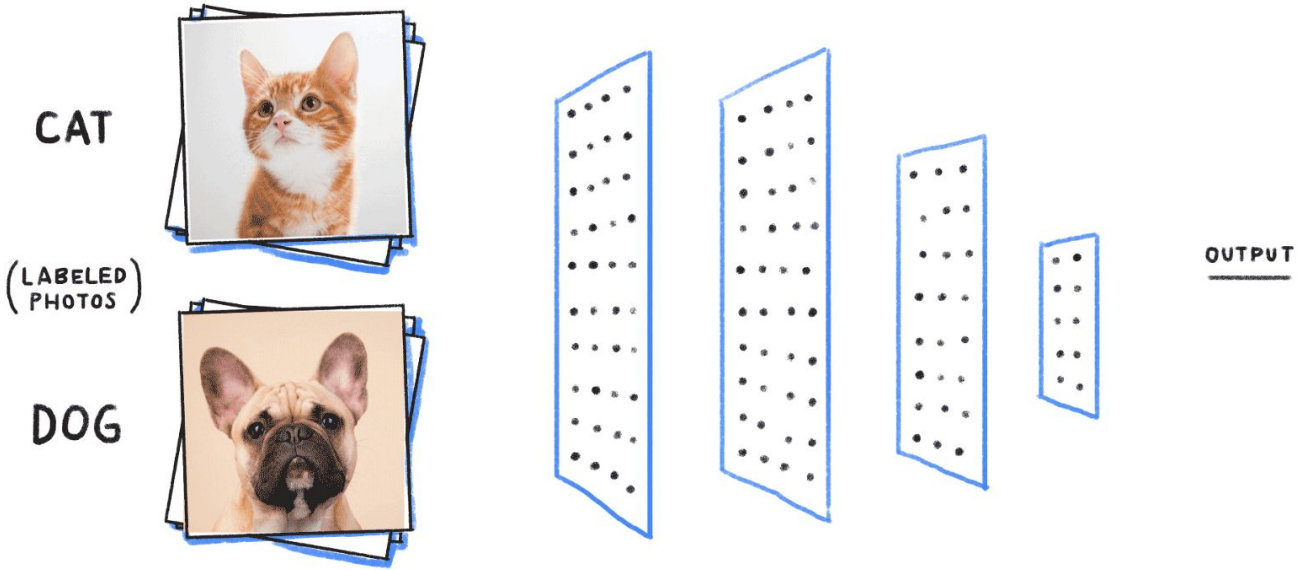

Kornblith et al., Do Better ImageNet Models Transfer Better?, arxiv 2018 https://arxiv.org/abs/1805.08974

3

# Same task different domain

# Image classification

We will introduce the Image Classification problem, which is the task of assigning an input image one label from a fixed set of categories.




source : https://becominghuman.ai/building-an-image-classifier-using-deep-learning-in-python-totally-from-a-beginners-perspective-be8dbaf22dd8

source : http://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html

# Example: Oxford 17 Flower Datasets

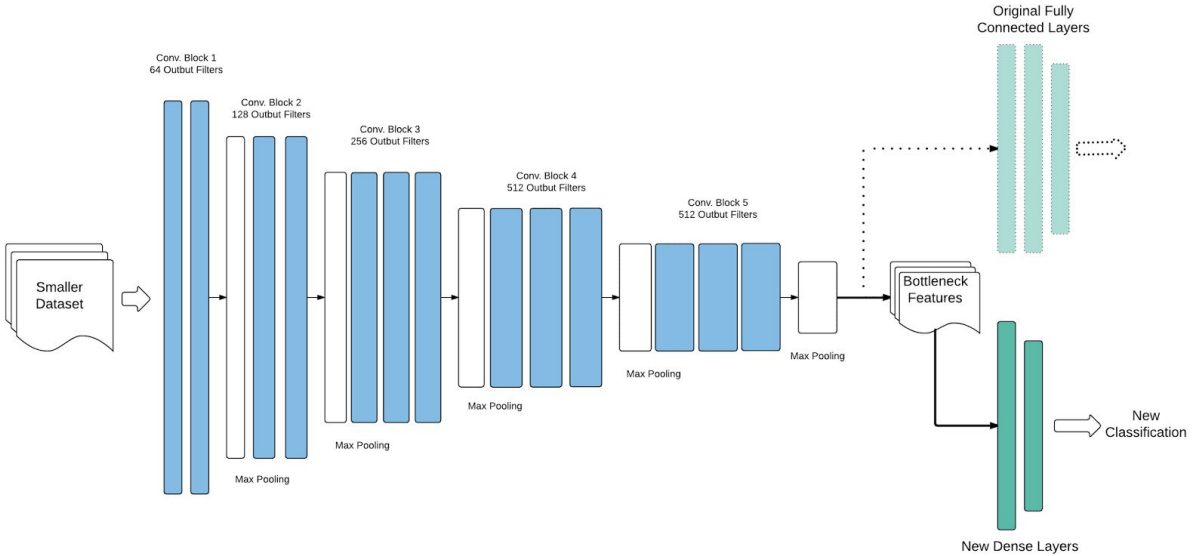
## Overview

- 17 category flower dataset with 80 images for each class.
- The flowers chosen are some common flowers in the UK.
- The images have large scale, pose and light variations and there are also classes with large variations of images within the class and close similarity to other classes.
- A subset of the images have been groundtruth labelled for segmentation.

## Class Examples

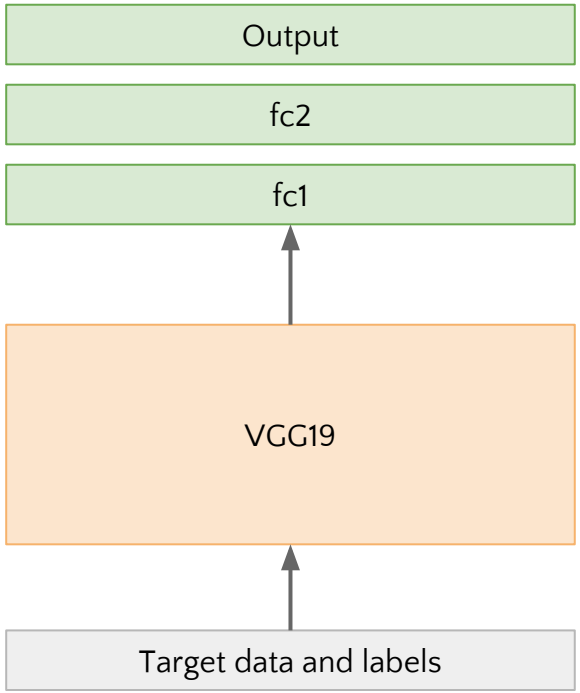
# VGG19

Karen Simonyan, Andrew Zisserman,
**Very Deep Convolutional Networks for Large-Scale Image Recognition,**
arXiv 2014 https://arxiv.org/abs/1409.1556



# Model 1: VGG19 (random initialized weights) + 2 Dense layers + Output layer

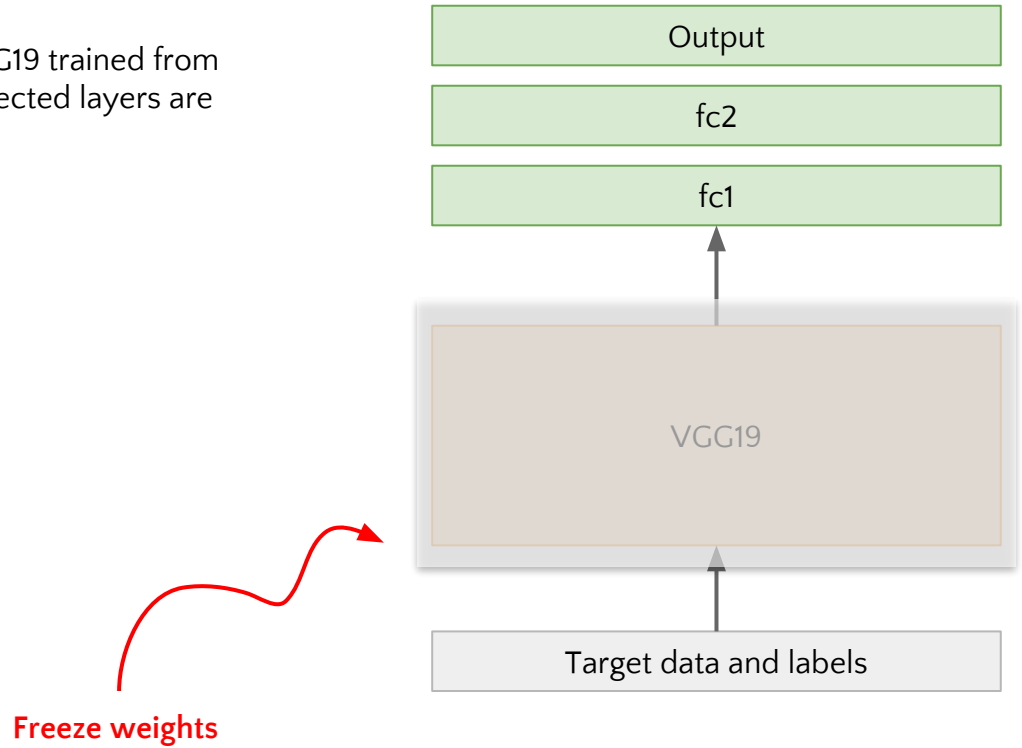
- The model is trained from scratch.
- no pretrained weights concept.



# Model 2: VGG (pre-trained weights) + 2 Dense layers + Output layer

- Use pretrained weights VGG19 trained from Imagenet, new 2 fully connected layers are initialized randomly.

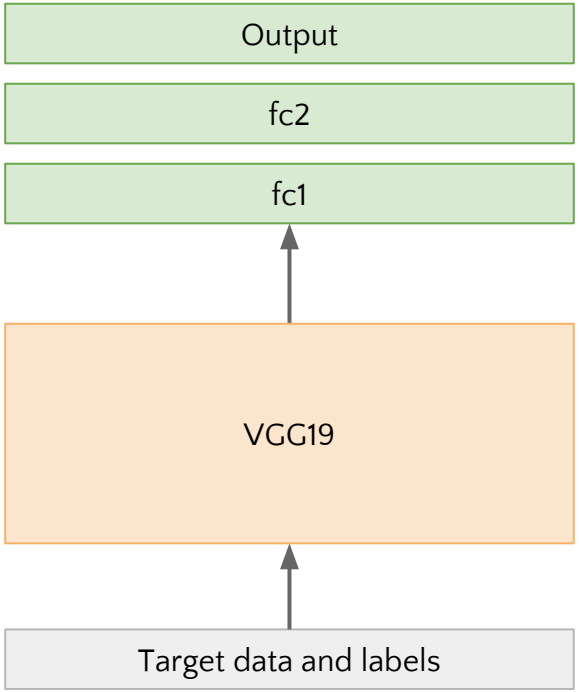
- Feature extractor concept.



# Model 3: VGG19 (pre-trained weights) + 2 Dense layers + Output layer

- The whole model, VGG19 and new fully connected layers, is trained to adapt with target task.

- Unfreeze all layers; fine-tune concept.



# Model 4: VGG (pre-trained weights) + 2 Dense layers + Output layer

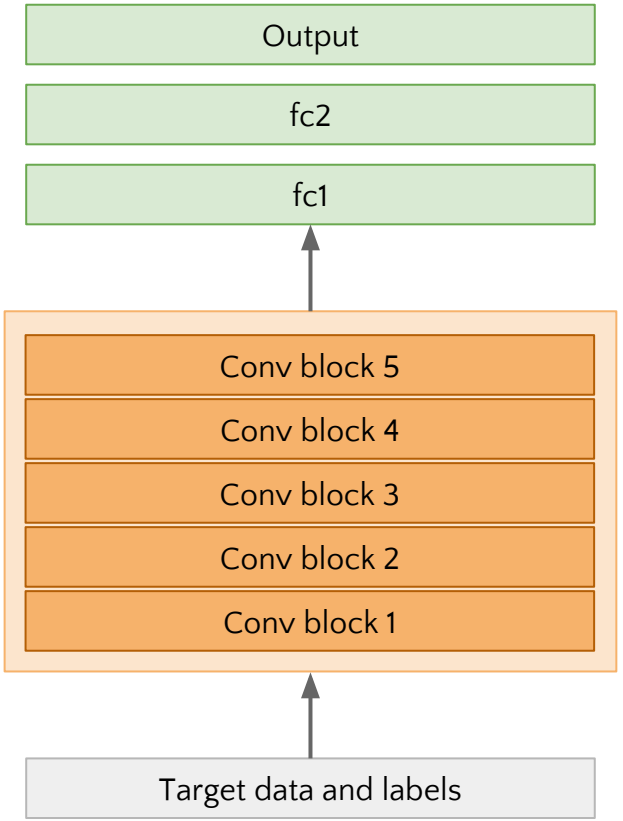
- Use "**chain-thaw**" fine-tune technique that sequentially unfreezes and fine-tune a single layer at a time to increases accuracy on the target task at the expense of extra computational power needed for the fine-tuning and would able to adjust the individual patterns across the network with a reduced risk of overfitting.

- Chain-thaw contains 3 sequential phases

  1. fine-tunes any new layers.

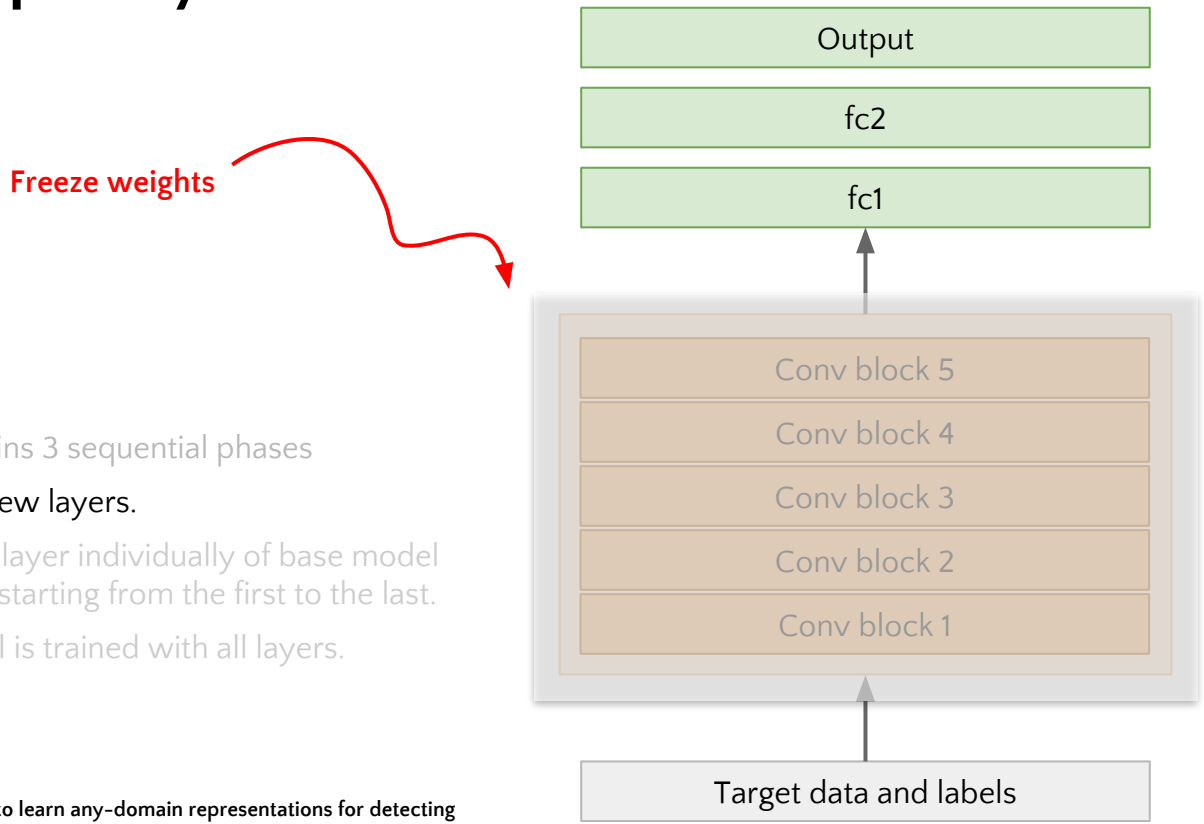
  2. fine-tunes each layer individually of base model (VGG in this case) starting from the first to the last.

  3. the entire model is trained with all layers.




Felbo et al., **Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm,** arxiv 2017, https://arxiv.org/pdf/1708.00524.pdf

# Model 4: VGG (pre-trained weights) + 2 Dense layers + Output layer




- Chain-thaw contains 3 sequential phases

1. fine-tunes any new layers.
2. fine-tunes each layer individually of base model (VGG in this case) starting from the first to the last.
3. the entire model is trained with all layers.

Felbo et al., **Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm,** arxiv 2017, https://arxiv.org/pdf/1708.00524.pdf

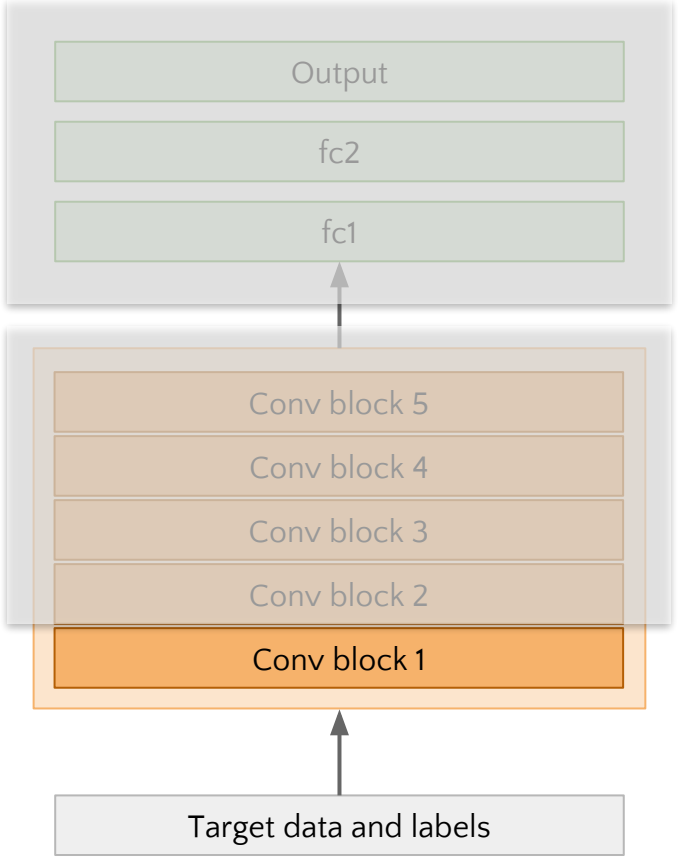
# Model 4: VGG (pre-trained weights) + 2 Dense layers + Output layer

- Chain-thaw contains 3 sequential phases

1. fine-tunes any new layers.

2. fine-tunes each layer individually of base model (VGG in this case) starting from the first to the last.

3. the entire model is trained with all layers.




Felbo et al., **Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm,** arxiv 2017, https://arxiv.org/pdf/1708.00524.pdf

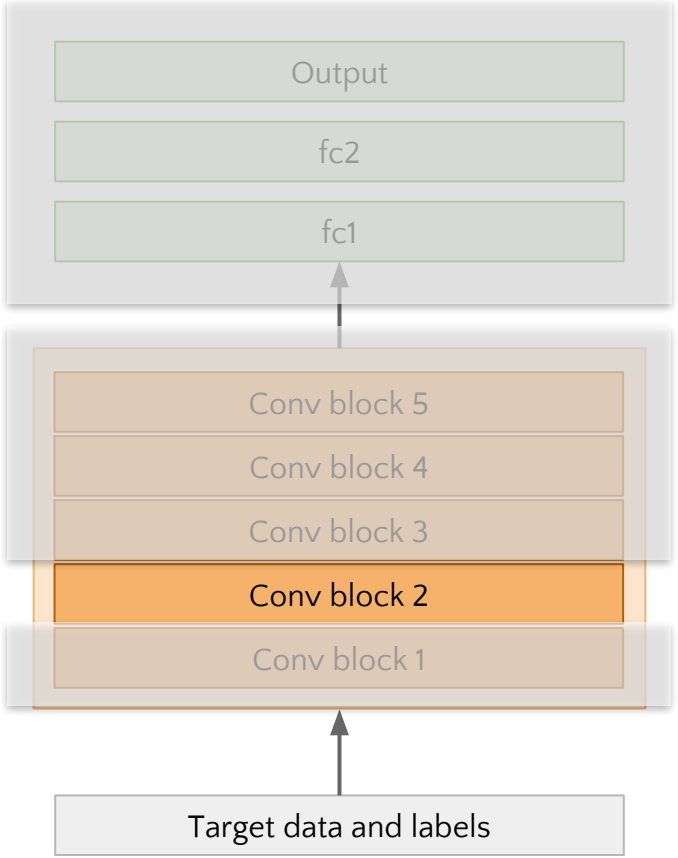
# Model 4: VGG (pre-trained weights) + 2 Dense layers + Output layer

- Chain-thaw contains 3 sequential phases

1. fine-tunes any new layers.

2. fine-tunes each layer individually of base model (VGG in this case) starting from the first to the last.

3. the entire model is trained with all layers.




Felbo et al., **Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm,** arxiv 2017, https://arxiv.org/pdf/1708.00524.pdf

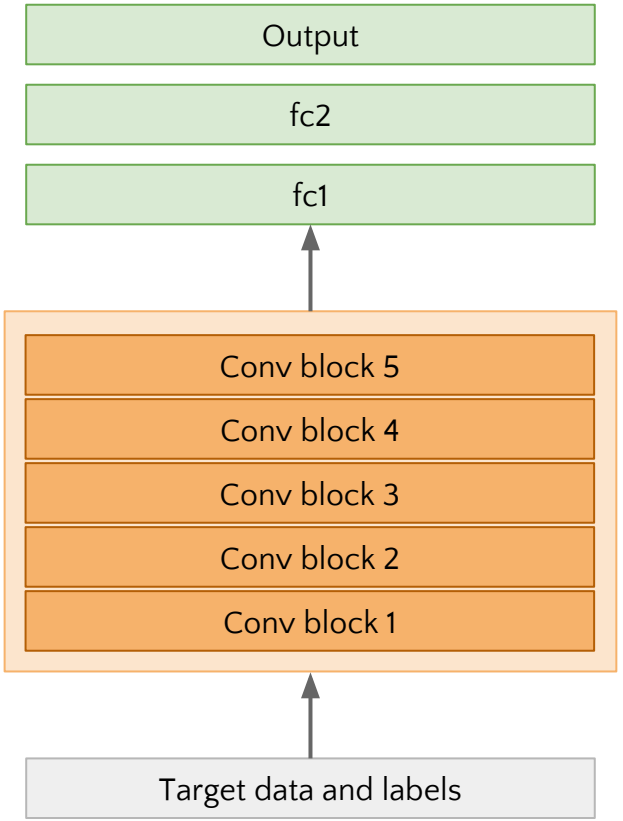
# Model 4: VGG (pre-trained weights) + 2 Dense layers + Output layer

- Chain-thaw contains 3 sequential phases

1. fine-tunes any new layers.

2. fine-tunes each layer individually of base model (VGG in this case) starting from the first to the last.

3. the entire model is trained with all layers.




Felbo et al., **Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm,** arxiv 2017, https://arxiv.org/pdf/1708.00524.pdf

4

# Different task

# Object Localization

An object localization model is similar to a classification model. But the trained localization model also predicts where the object is located in the image by drawing a bounding box around it. For example, dogs is located in the image below. The information of the bounding box containing (xmin, ymin, xmax, ymax) is also included in the model output.

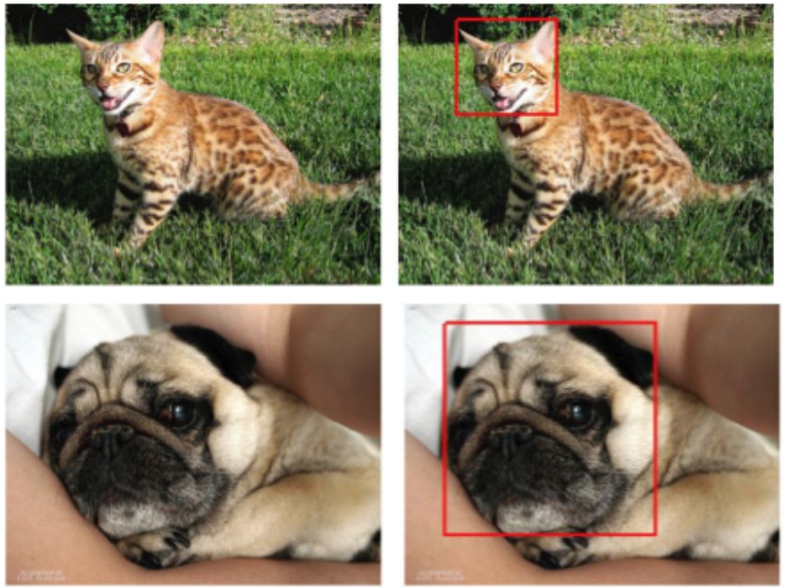

source : https://towardsdatascience.com/real-time-object-detection-with-tensorflow-detection-model-e7fd20421d5d

# Example: Oxford-IIIT Pet Datasets

## Overview

- 37 category pet dataset with roughly 200 images for each class.
- The images have a large variations in scale, pose and lighting.
- All images have an associated ground truth annotation of breed, head ROI, and pixel level trimap segmentation.

## Annotation Examples

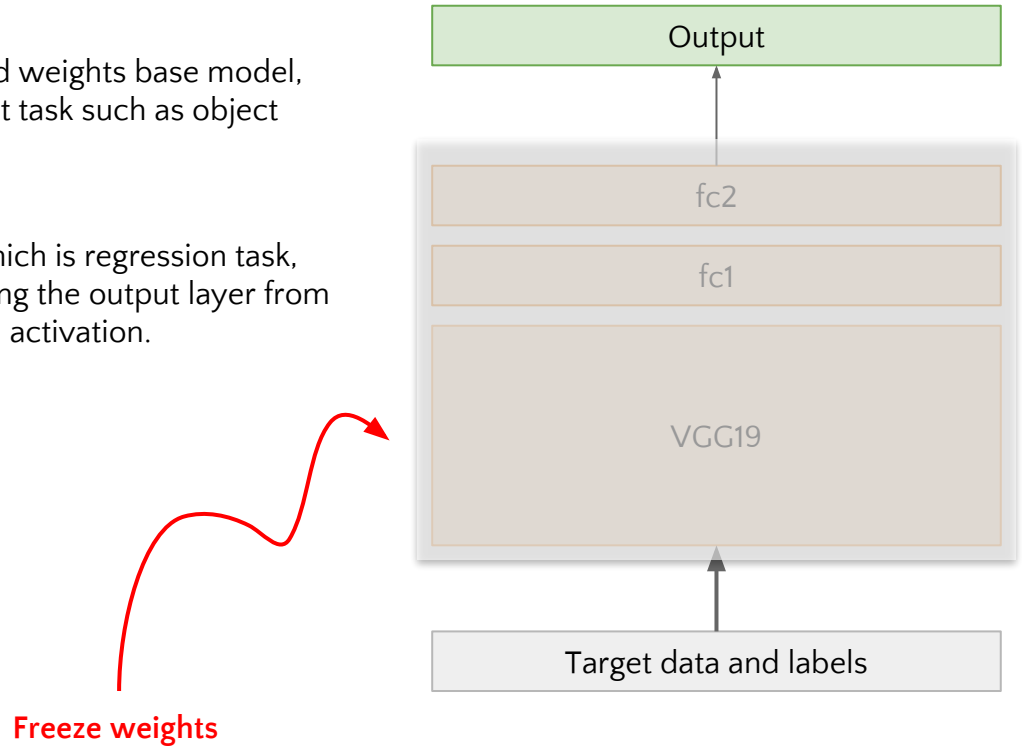


source : http://www.robots.ox.ac.uk/~vgg/data/pets/

# Model 1: VGG19 (pre-trained weights) + Output layer

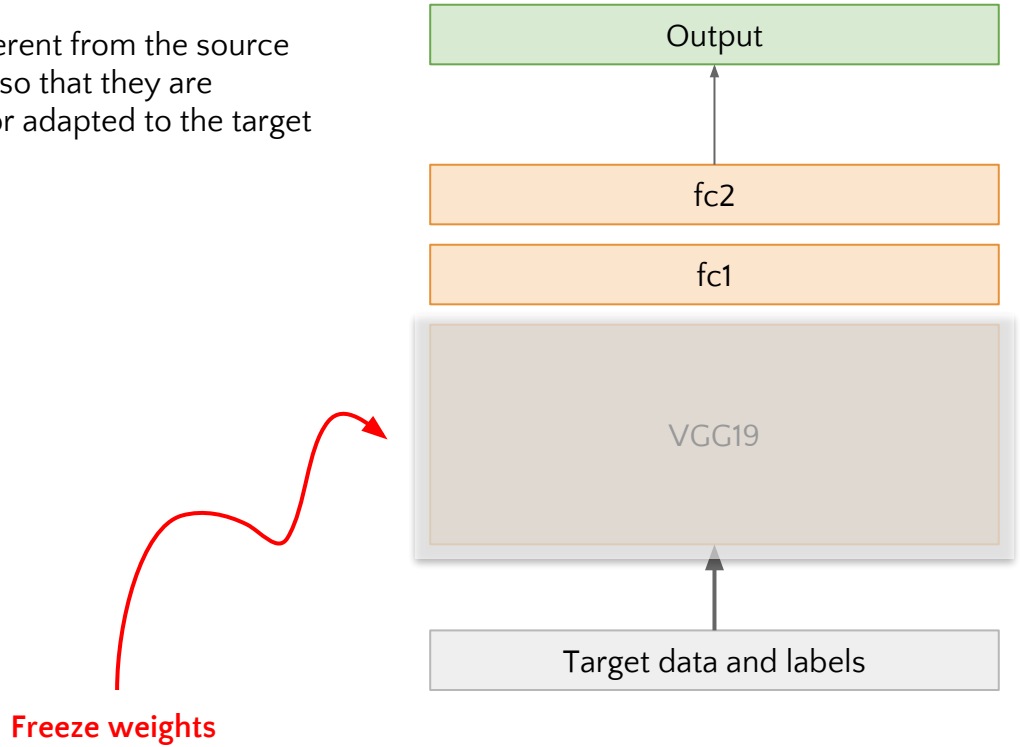
- You can also use pretrained weights base model, such as VGG19, for different task such as object localization.

- Object localization task, which is regression task, can be achieved by changing the output layer from softmax to be linear or relu activation.



# Model 2: VGG (pre-trained weights) + Output layer with Adaptation

- The tasks that are very different from the source task are harder to transfer, so that they are required to be fine-tuned or adapted to the target tasks.



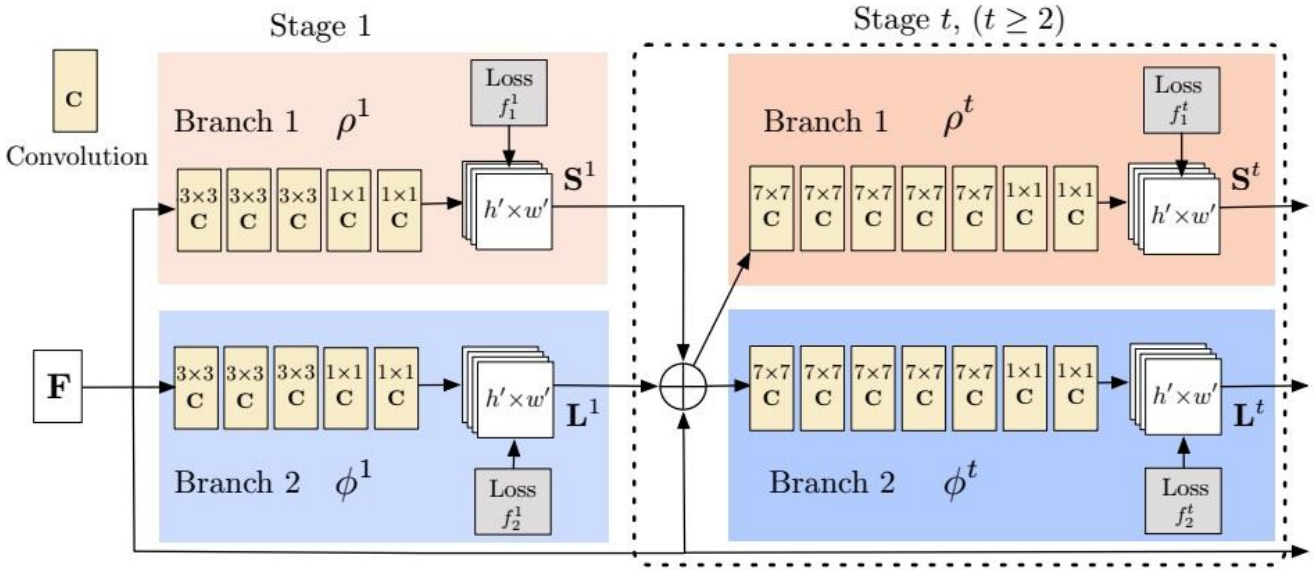
# More complex task: Pose Estimation

Human 2D pose estimation is the problem of localizing anatomical keypoints or “parts” having largely focused on finding body parts of individuals. Inferring the pose of multiple people in images, especially socially engaged individuals, presents a unique set of challenges.


Cao et al., **Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,** CVPR 2017 https://arxiv.org/abs/1611.08050

# More complex task: Pose Estimation

Tasks that are very different from source task are harder to transfer and also required much more layer up top.




Cao et al., **Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,** CVPR 2017 https://arxiv.org/abs/1611.08050

5

# Why adaptation is so important

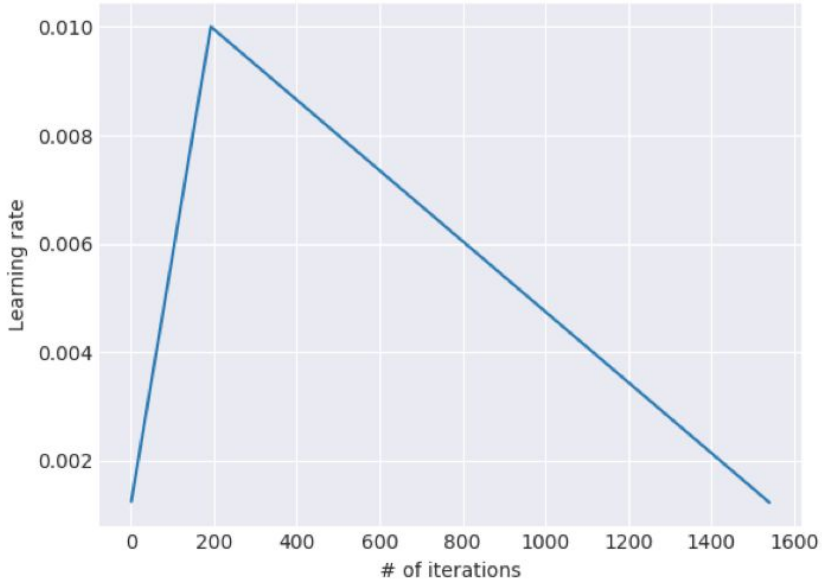
# Additional fine tuning techniques

**Slanted triangular learning rates (STLR)**, which first linearly increases the learning rate and then linearly decays it according to the following update schedule.

$$cut = \lfloor T \cdot cut\_frac \rfloor$$

$$p = \begin{cases} t/cut, & \text{if } t < cut \\ 1 - \frac{t - cut}{cut \cdot (1/cut\_frac - 1)}, & \text{otherwise} \end{cases}$$

$$\eta_t = \eta_{max} \cdot \frac{1 + p \cdot (ratio - 1)}{ratio}$$



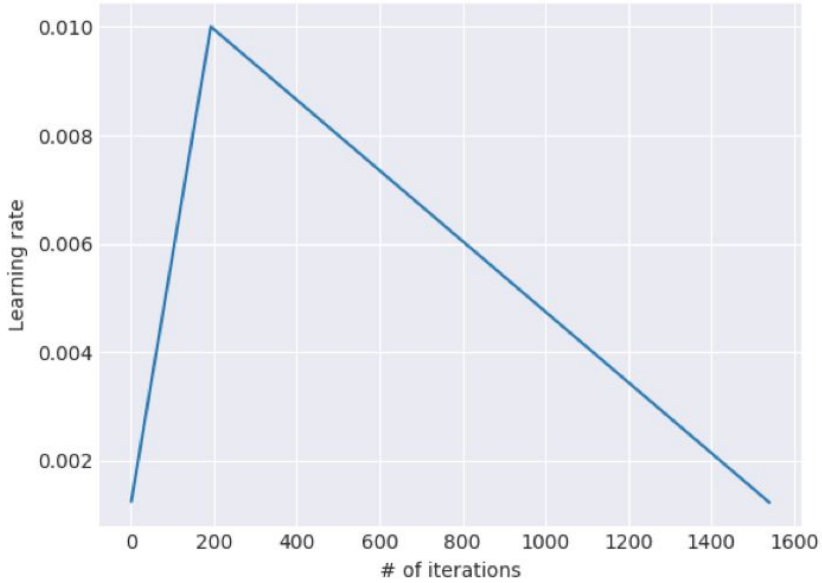

**Discriminative fine-tuning**

Instead of using the same learning rate for all layers of the model, discriminative fine-tuning allows us to tune each layer with different learning rates.

Jeremy Howard and Sebastian Ruder, **Universal Language Model Fine-tuning for Text Classification,** ACL 2018 https://arxiv.org/pdf/1801.06146.pdf

# Additional fine tuning techniques (cont.)

"Empirically found it to work well"




Jeremy Howard and Sebastian Ruder, **Universal Language Model Fine-tuning for Text Classification,** ACL 2018 https://arxiv.org/pdf/1801.06146.pdf

# Conclusion

- Possible to train very large models on small data by using transfer learning and domain adaptation.
- Off the shelf features, as fixed features, work very well in various domains and tasks for smaller samples, but should be adapted to increase performance (fine tune).
- Learned features are hierarchical representations, different layers learn at different level and transfer differently, Lower layers of network contain very generic features, higher layers more task specific features. Tasks that are very different are harder to transfer and required more layer up top.
- Supervised domain adaptation via fine tuning almost always improves performance.
- Best backbone for transfer overall is ResNet family.

# References

Lecture slide:

https://www.slideshare.net/xavigiro/transfer-learning-d2l4-insightdcu-machine-learning-workshop-2017?qid=e5de4681-9045-4ad8-b88c-b11be4aba7a7&v=&b=&from_search=1