18AIC301J: DEEP LEARNING TECHNIQUES

B. Tech in ARTIFICIAL INTELLIGENCE, 5th semester

Faculty: **Dr. Athira Nambiar**

Section: A, slot:D

Venue: TP 804

Academic Year: 2022-22

UNIT-4

DenseNet Architecture, Transfer Learning
Need for Transfer Learning, Deep Transfer Learning, Types of Deep Transfer learning, Applications of Transfer learning
Transfer learning implementation using VGG16 model to classify images
Sequence Learning Problems, Recurrent Neural Networks
Backpropagation through time, Unfolded RNN, The problem of exploding and vanishing Gradients, Seq to Seq Models
Building a RNN to perform Character level language modeling.
How gates help to solve the problem of vanishing gradients, Long-Short Term Memory architectures
Dealing with exploding gradients, Gated Recurrent Units, Introduction to Encoder Decoder Models, Applications of Encoder Decoder Models
Build a LSTM network for Named Entity recognition.

UNIT-4

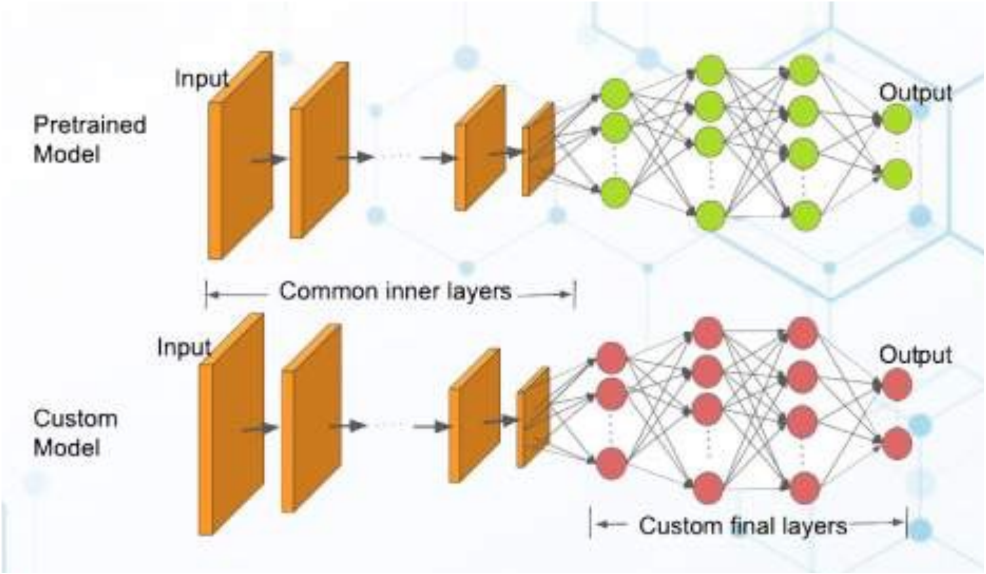
DenseNet Architecture, Transfer Learning
Need for Transfer Learning, Deep Transfer Learning, Types of Deep Transfer learning, Applications of Transfer learning
Transfer learning implementation using VGG16 model to classify images
Sequence Learning Problems, Recurrent Neural Networks
Backpropagation through time, Unfolded RNN, The problem of exploding and vanishing Gradients, Seq to Seq Models
Building a RNN to perform Character level language modeling.
How gates help to solve the problem of vanishing gradients, Long-Short Term Memory architectures
Dealing with exploding gradients, Gated Recurrent Units, Introduction to Encoder Decoder Models, Applications of Encoder Decoder Models
Build a LSTM network for Named Entity recognition.

Transfer Learning

“You need a lot of a data if you want to train/use CNNs”

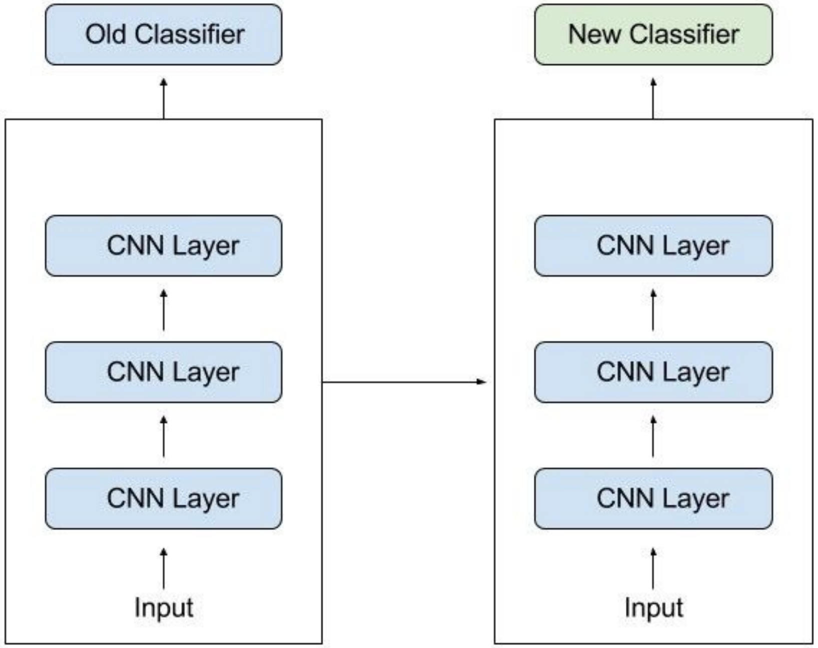
TRANSFER LEARNING

- ***The reuse of a pre-trained model on a new problem is known as transfer learning in machine learning.***
- ***A machine uses the knowledge learned from a prior assignment to increase prediction about a new task in transfer learning.***



TRANSFER LEARNING

- Transfer learning is a machine learning technique in which an AI that has been trained to perform a specific task is being reused (repurposed) as a starting point for another similar task.
- Transfer learning is widely used since starting from a pre-trained AI model can dramatically reduce the computational time required if training is performed from scratch.



TRANSFER LEARNING

Transfer learning offers a number of advantages, the most important of which are

(i) reduced training time,

(ii) improved neural network performance (in most circumstances),

(iii) the absence of a large amount of data.

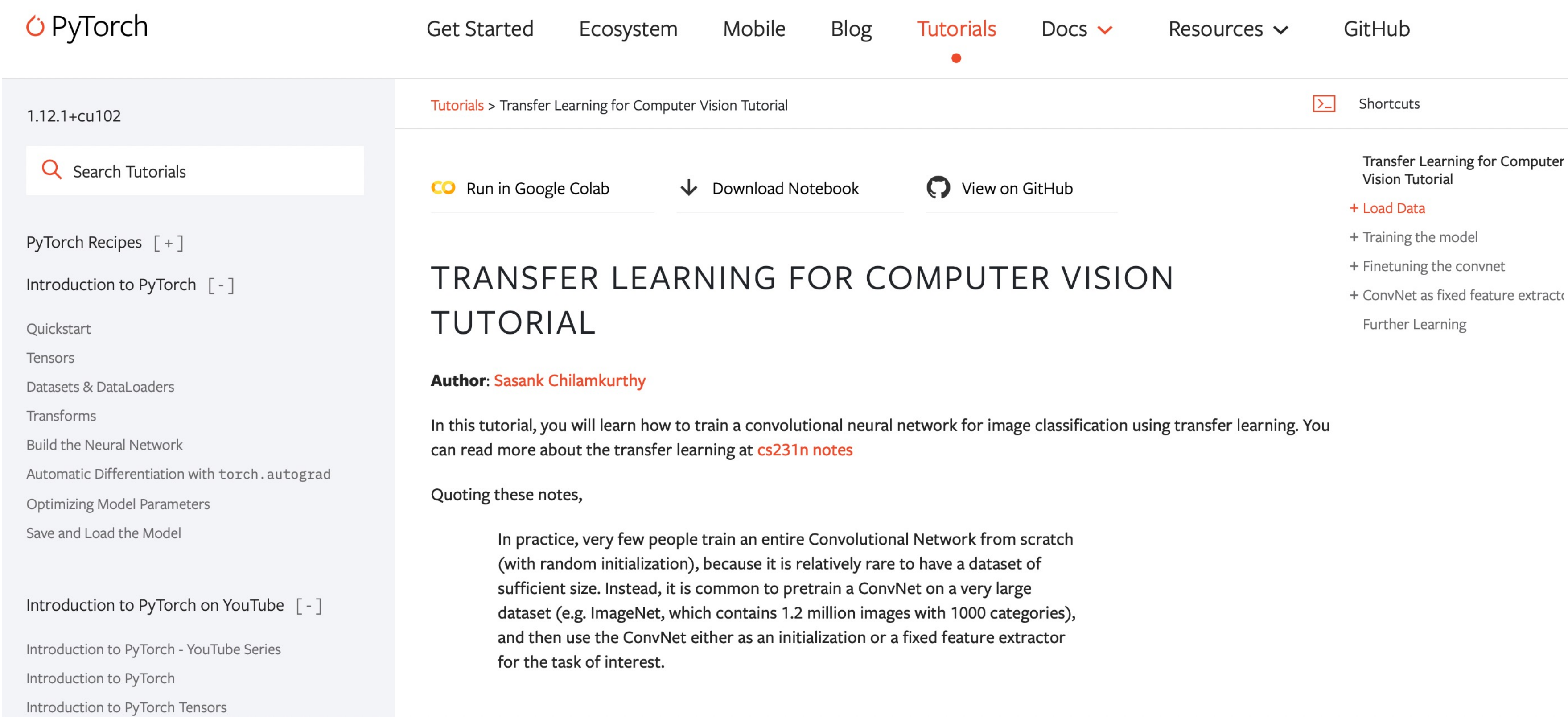
To train a neural model from scratch, a lot of data is typically needed, but access to that data isn't always possible – this is when transfer learning comes in handy.

TRANSFER LEARNING

When to Use Transfer Learning?

When we don’t have enough annotated data to train our model with.
When there is a pre-trained model that has been trained on similar data and tasks.

TRANSFER LEARNING



PyTorch

Get Started | Ecosystem | Mobile | Blog | Tutorials | Docs | Resources | GitHub

1.12.1+cu102

Search Tutorials

PyTorch Recipes [+]

Introduction to PyTorch [-]

Quickstart
Tensors
Datasets & DataLoaders
Transforms
Build the Neural Network
Automatic Differentiation with `torch.autograd`
Optimizing Model Parameters
Save and Load the Model

Introduction to PyTorch on YouTube [-]

Introduction to PyTorch - YouTube Series
Introduction to PyTorch
Introduction to PyTorch Tensors

Tutorials > Transfer Learning for Computer Vision Tutorial

Shortcuts

Run in Google Colab | Download Notebook | View on GitHub

TRANSFER LEARNING FOR COMPUTER VISION TUTORIAL

Author: Sasank Chilamkurthy

In this tutorial, you will learn how to train a convolutional neural network for image classification using transfer learning. You can read more about the transfer learning at cs231n notes

Quoting these notes,

> In practice, very few people train an entire Convolutional Network from scratch (with random initialization), because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest.

Transfer Learning for Computer Vision Tutorial
+ Load Data
+ Training the model
+ Finetuning the convnet
+ ConvNet as fixed feature extractc
Further Learning

https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

TRANSFER LEARNING

These two major transfer learning scenarios look as follows:

- **Finetuning the convnet**: Instead of random initialization, we initialize the network with a pretrained network, like the one that is trained on imagenet 1000 dataset. Rest of the training looks as usual.
- **ConvNet as fixed feature extractor**: Here, we will freeze the weights for all of the network except that of the final fully connected layer. This last fully connected layer is replaced with a new one with random weights and only this layer is trained.

https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

TRANSFER LEARNING

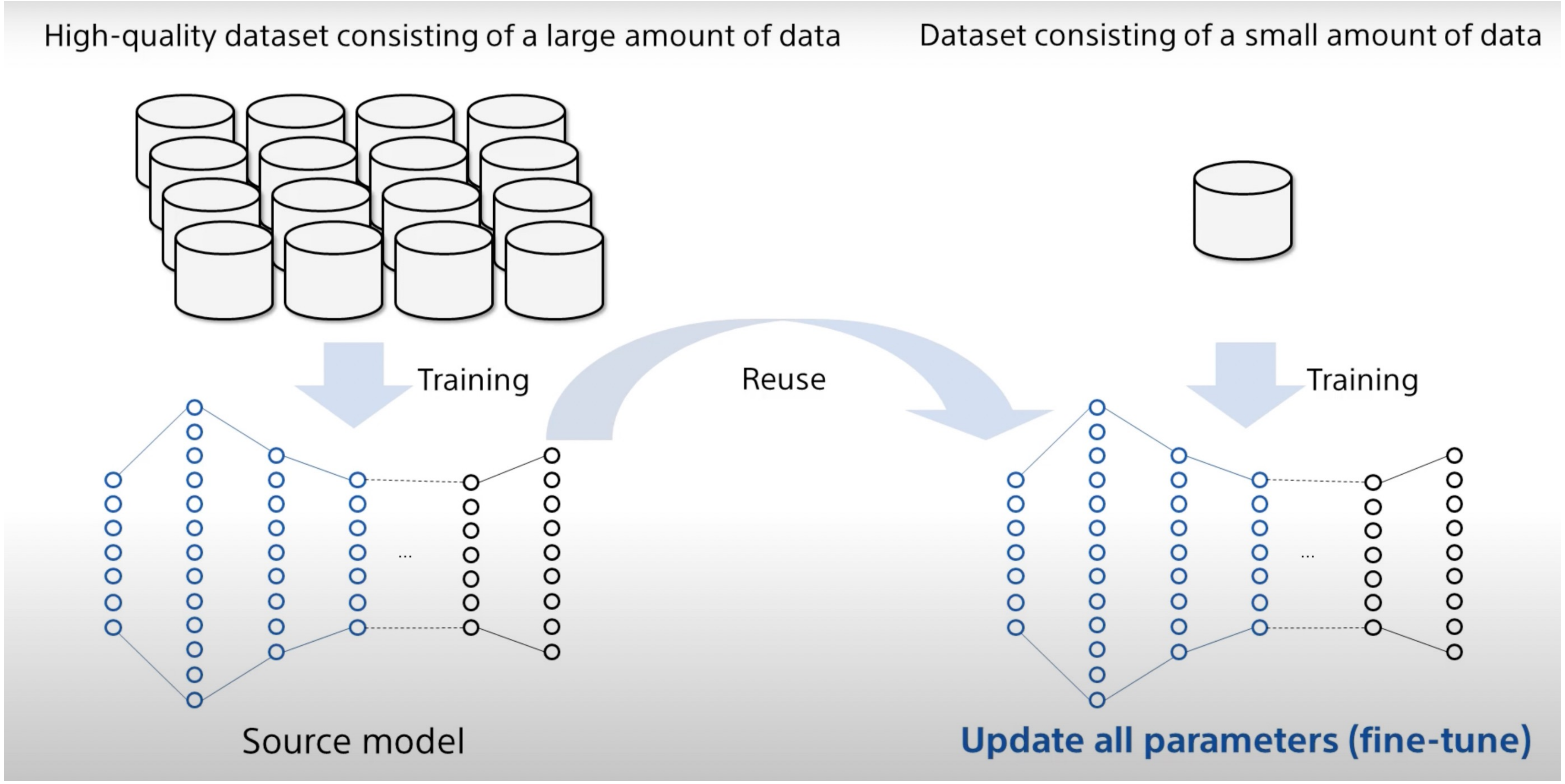
Finetuning the convnet

Load a pretrained model and reset final fully connected layer.

```
model_ft = models.resnet18(pretrained=True)
num_ftrs = model_ft.fc.in_features
# Here the size of each output sample is set to 2.
# Alternatively, it can be generalized to nn.Linear(num_ftrs, len(class_names)).
model_ft.fc = nn.Linear(num_ftrs, 2)

model_ft = model_ft.to(device)

criterion = nn.CrossEntropyLoss()

# Observe that all parameters are being optimized
optimizer_ft = optim.SGD(model_ft.parameters(), lr=0.001, momentum=0.9)

# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_ft, step_size=7, gamma=0.1)
```

https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html

TRANSFER LEARNING



TRANSFER LEARNING

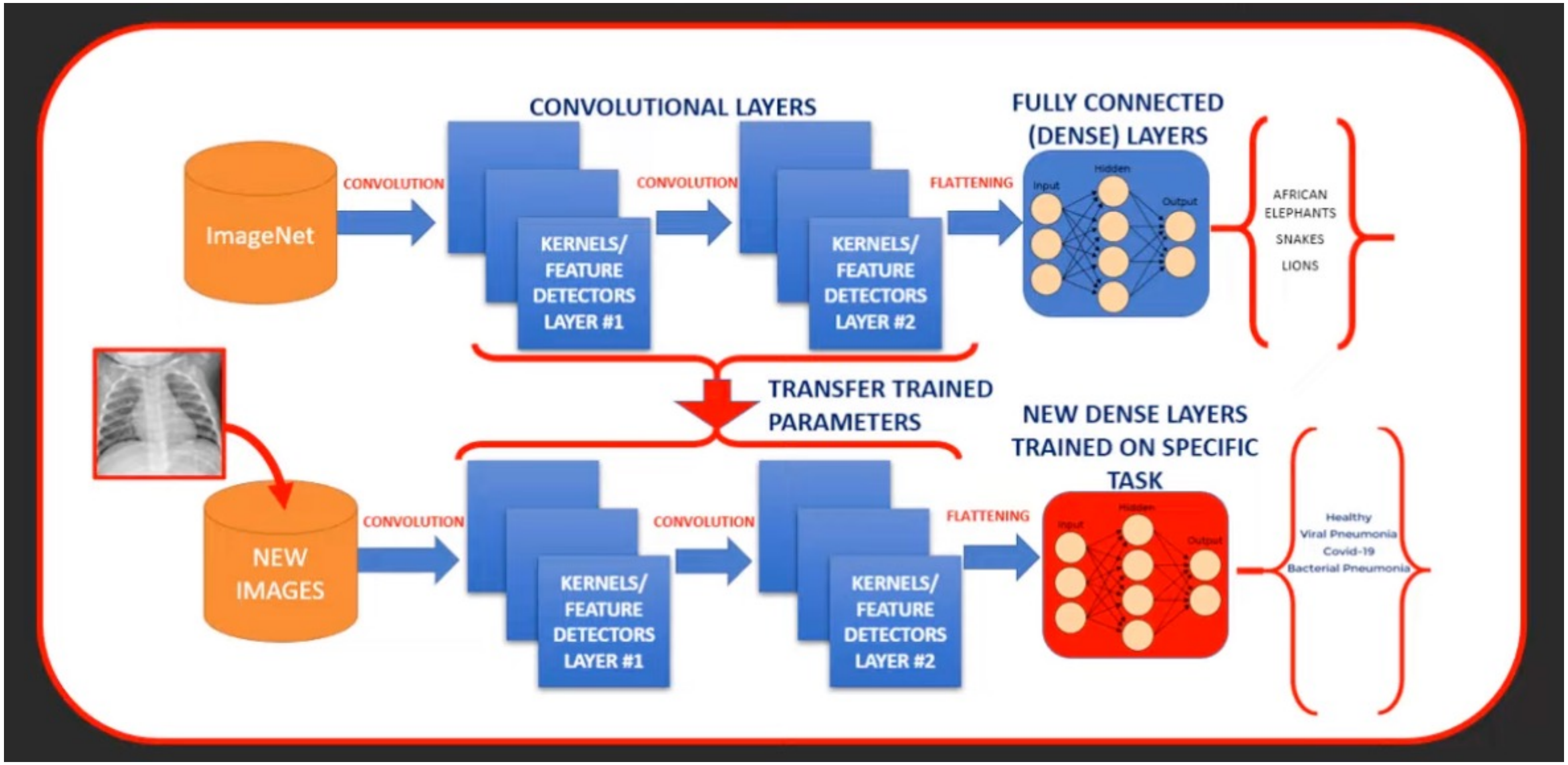
ConvNet as fixed feature extractor

Here, we need to freeze all the network except the final layer. We need to set `requires_grad = False` to freeze the parameters so that the gradients are not computed in `backward()`.

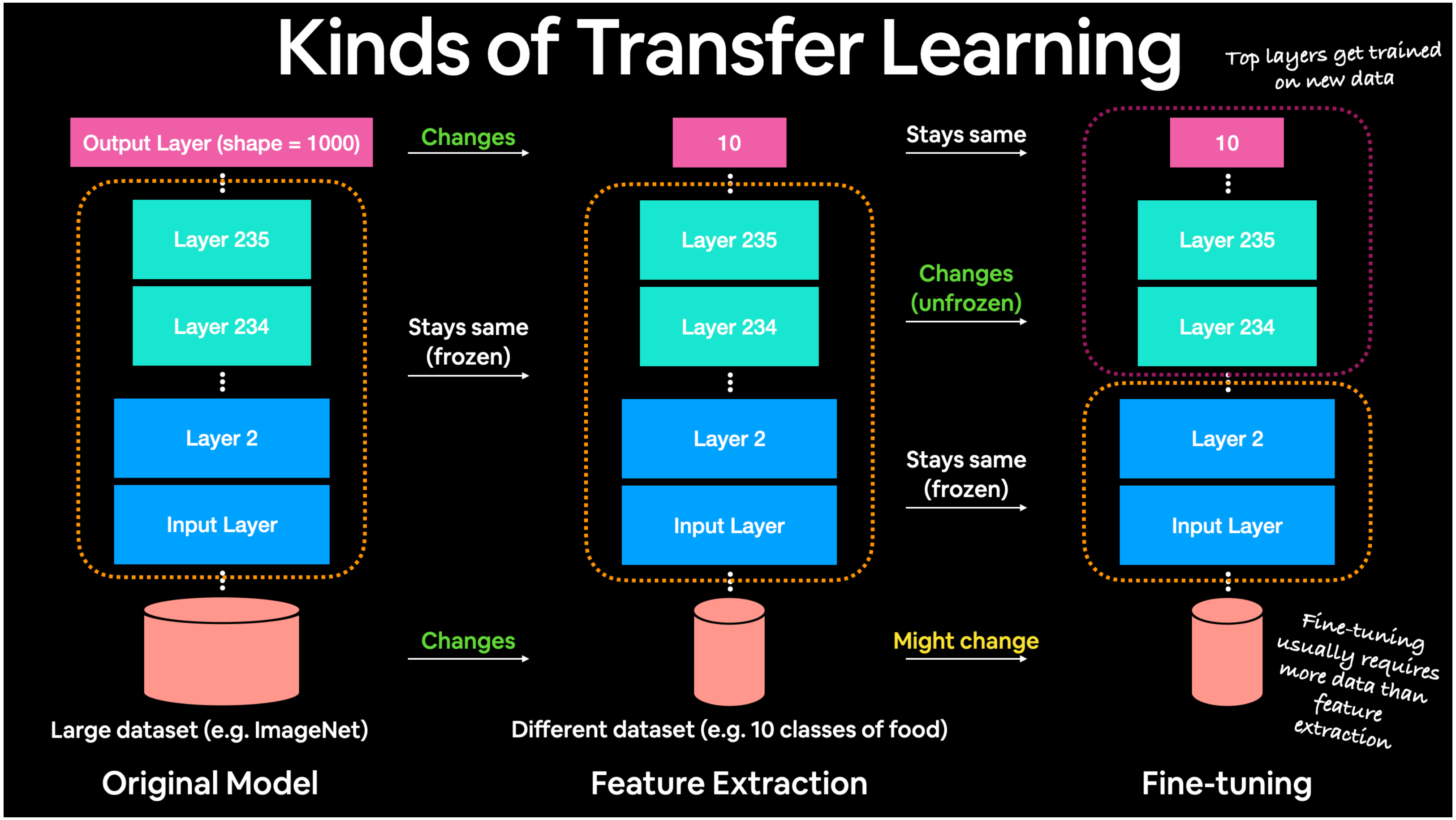
You can read more about this in the documentation here.

```
model_conv = torchvision.models.resnet18(pretrained=True)
for param in model_conv.parameters():
    param.requires_grad = False

# Parameters of newly constructed modules have requires_grad=True by default
num_ftrs = model_conv.fc.in_features
model_conv.fc = nn.Linear(num_ftrs, 2)

model_conv = model_conv.to(device)

criterion = nn.CrossEntropyLoss()

# Observe that only parameters of final layer are being optimized as
# opposed to before.
optimizer_conv = optim.SGD(model_conv.fc.parameters(), lr=0.001, momentum=0.9)

# Decay LR by a factor of 0.1 every 7 epochs
exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size=7, gamma=0.1)
```

TRANSFER LEARNING



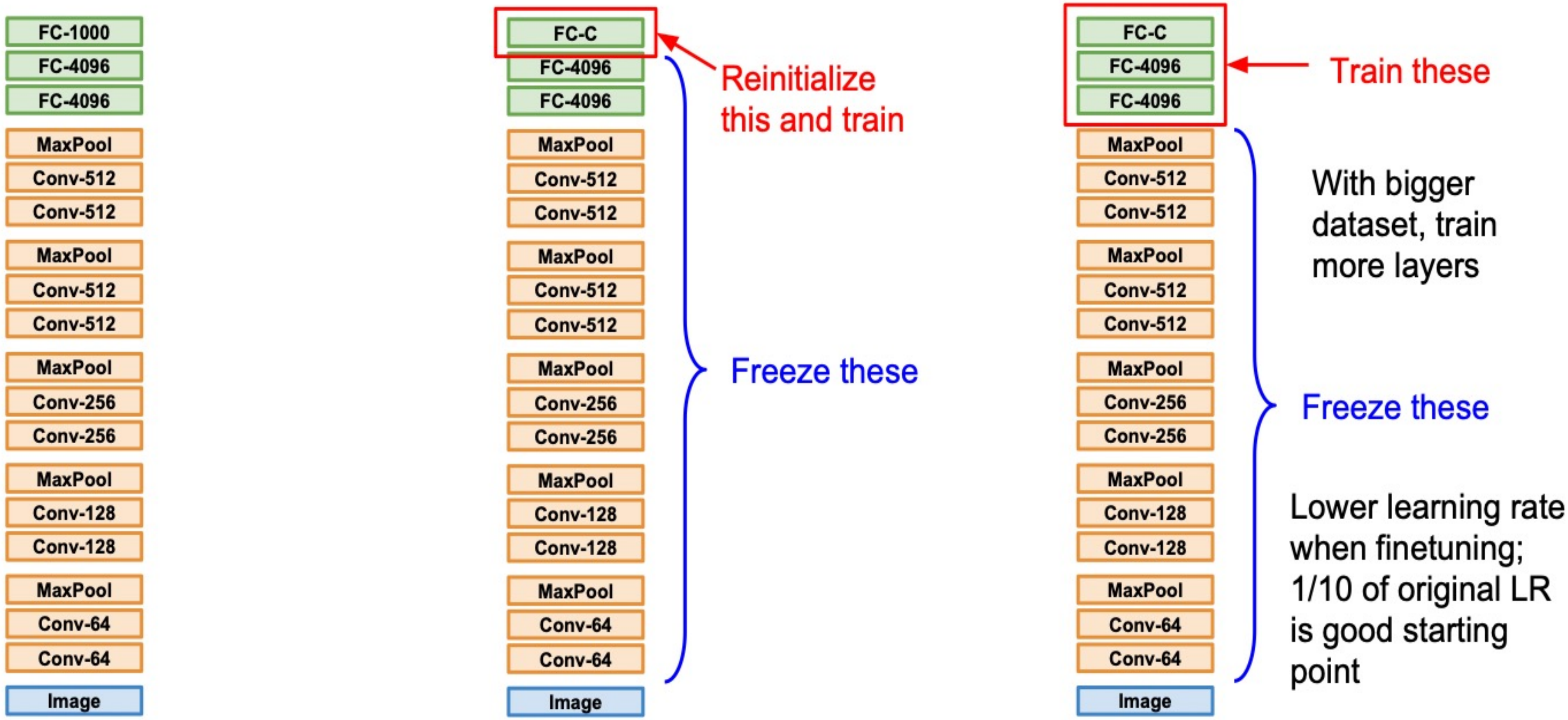
TRANSFER LEARNING

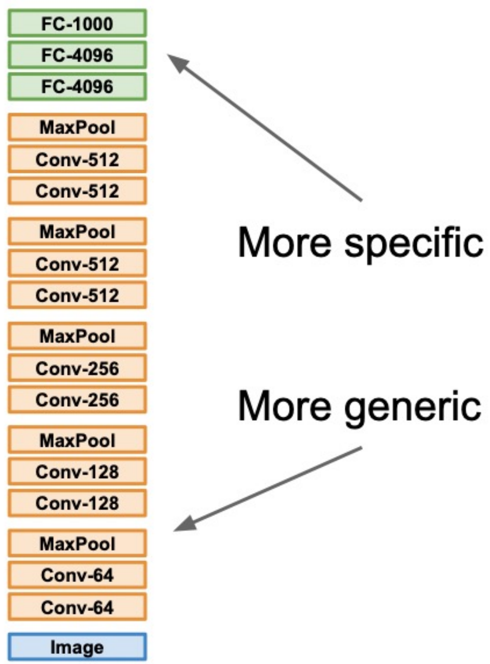


Transfer Learning with CNNs

Donahue et al, “DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition”, ICML 2014
Razavian et al, “CNN Features Off-the-Shelf: An Astounding Baseline for Recognition”, CVPR Workshops 2014

1. Train on Imagenet
2. Small Dataset (C classes)
3. Bigger dataset





	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers

TRANSFER LEARNING

Which to use?

There are two main factors that will affect your choice of approach:

1.Your dataset size

2.Similarity of your dataset to the pre-trained dataset (typically ImageNet)

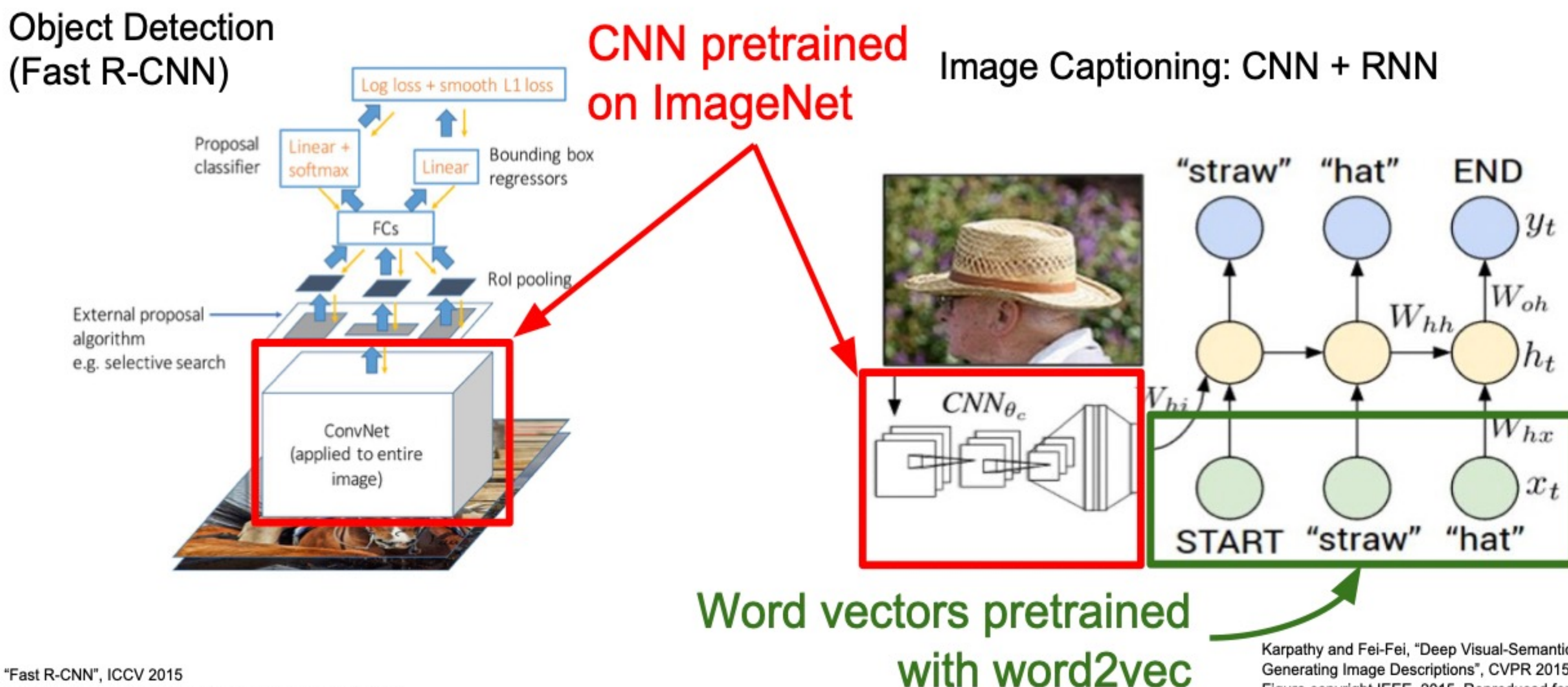
	Similar dataset	Different dataset
Small dataset	Transfer learning: highest level features + classifier	Transfer learning: lower level features + classifier
Large dataset	Fine-tune*	Fine-tune*

https://medium.com/deeplearningsandbox/how-to-use-transfer-learning-and-fine-tuning-in-keras-and-tensorflow-to-build-an-image-recognition-94b0b02444f2

TRANSFER LEARNING

Transfer learning with CNNs is pervasive…

(it's the norm, not an exception)



Girshick, "Fast R-CNN", ICCV 2015
Figure copyright Ross Girshick, 2015. Reproduced with permission.

Karpathy and Fei-Fei, "Deep Visual-Semantic Alignments for Generating Image Descriptions", CVPR 2015
Figure copyright IEEE, 2015. Reproduced for educational purposes.

https://medium.com/deeplearningsandbox/how-to-use-transfer-learning-and-fine-tuning-in-keras-and-tensorflow-to-build-an-image-recognition-94b0b02444f2

TRANSFER LEARNING

Takeaway for your projects and beyond:

Have some dataset of interest but it has < ~1M images?

1. Find a very large dataset that has similar data, train a big ConvNet there
2. Transfer learn to your dataset

Deep learning frameworks provide a "Model Zoo" of pretrained models so you don't need to train your own

Caffe: https://github.com/BVLC/caffe/wiki/Model-Zoo
TensorFlow: https://github.com/tensorflow/models
PyTorch: https://github.com/pytorch/vision

https://medium.com/deeplearningsandbox/how-to-use-transfer-learning-and-fine-tuning-in-keras-and-tensorflow-to-build-an-image-recognition-94b0b02444f2

Learning Resources

- Charu C. Aggarwal, Neural Networks and Deep Learning, Springer, 2018.
- Eugene Charniak, Introduction to Deep Learning, MIT Press, 2018.
- Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016.
- Michael Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.
- Deng & Yu, Deep Learning: Methods and Applications, Now Publishers, 2013.
- https://medium.com/deeplearningsandbox/how-to-use-transfer-learning-and-fine-tuning-in-keras-and-tensorflow-to-build-an-image-recognition-94b0b02444f2
- https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html
- Lecture 7 | Training Neural Networks II (Stanford)https://youtu.be/_JB0AO7QxSA?t=4212
- http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture7.pdf

Thank you