Week 5 – Lesson 1:
Training CNNs on Small Datasets

Dr. Hongping Cai

Department of Computer Science

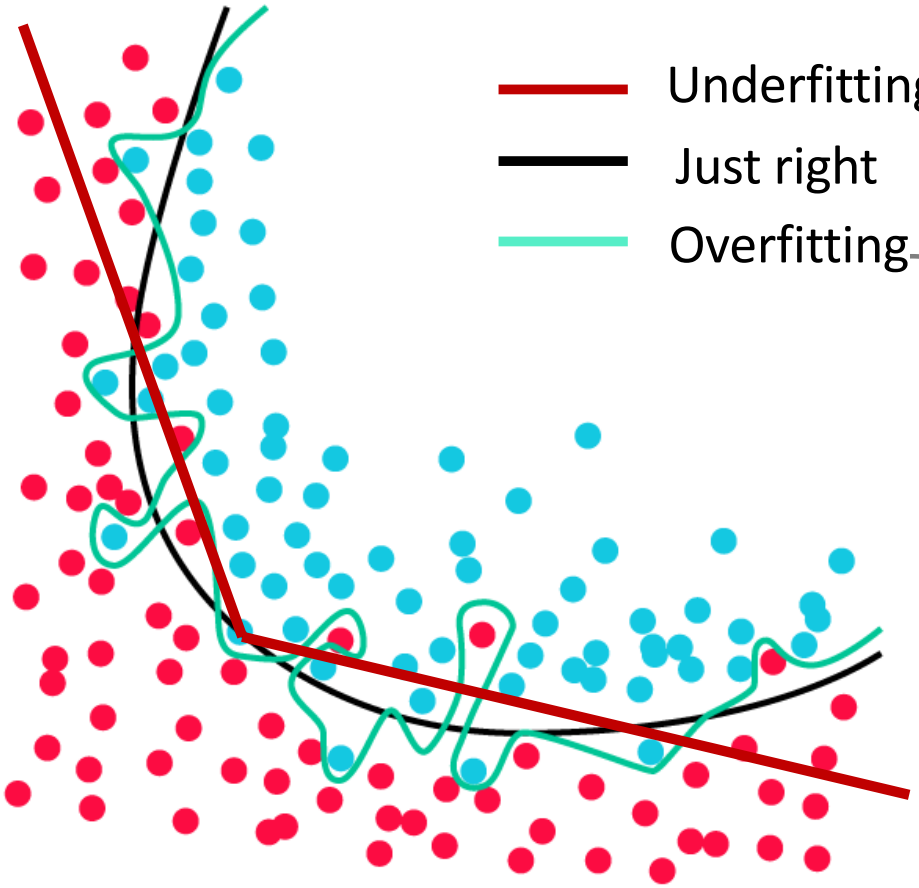
University of Bath

Topic 1:
Data Augmentation

Training CNNs

- CNN is trained in the same way like MLP: gradient descent with backpropagation.
- In practice, it is relatively rare to have sufficient training data:
  - Data collection and annotation is expensive.
  - Sometimes near-impossible (e.g., medical images of a rare disease)
- Therefore, overfitting is always the main concern.

How to Prevent Underfitting and Overfitting



Underfitting

Just right

Overfitting

- Complexify model
- Add more nodes/layers
- Train longer

- More training data
  - Data augmentation
- Simplify model
- Regularization

Data Augmentation

- **Data augmentation** is to generate synthetic data from existing training examples, by *augmenting* the samples via a number of random transformations.

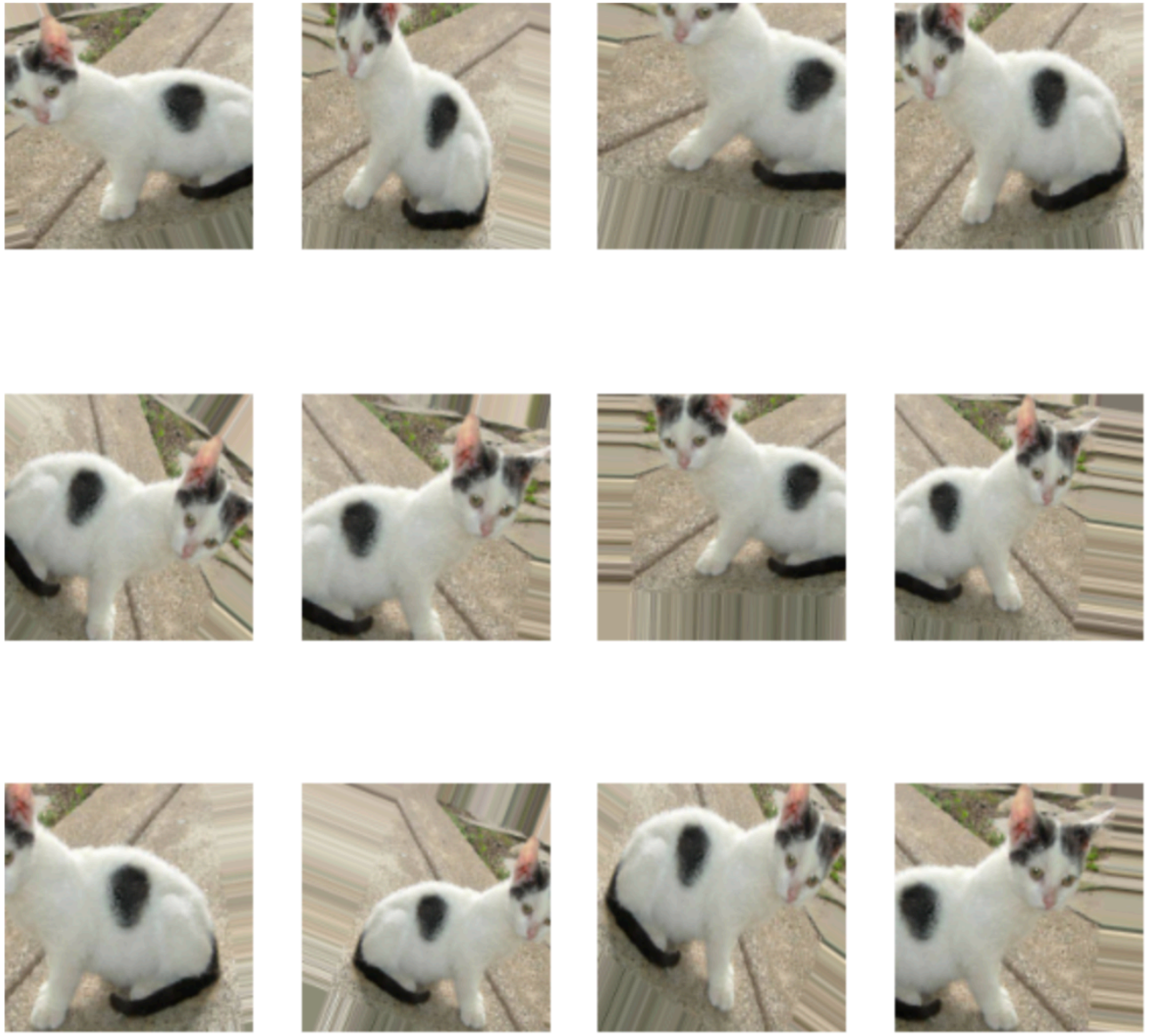
```
from keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range = 40,
        width_shift_range = 0.2,
        height_shift_range = 0.2,
        shear_range = 0.2,
        zoom_range = 0.2,
        horizontal_flip = True,
        fill_mode = 'nearest')

test_datagen = ImageDataGenerator(rescale=1./255)
```

The validation and test data shouldn't be augmented.

Augmented examples

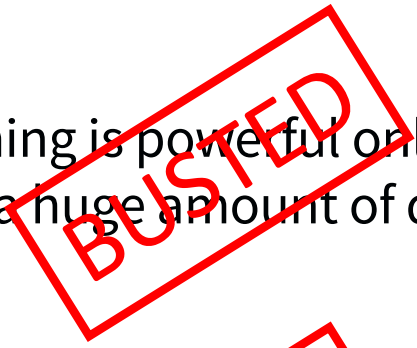


Reference for Topic 1

- Book: Francois Chollet. Deep Learning with Python. Manning. 2018.
- Blog by Francois Chollet: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

Topic 2:
Transfer Learning

“Deep learning is powerful only when you have a huge amount of data.”



“You can only use small CNNs if your dataset is small.”

Transfer learning: Making Less Data Cool Again

Transfer learning

- **Transfer learning** is a machine learning technique where a model trained on one task is re-purposed on another related task.

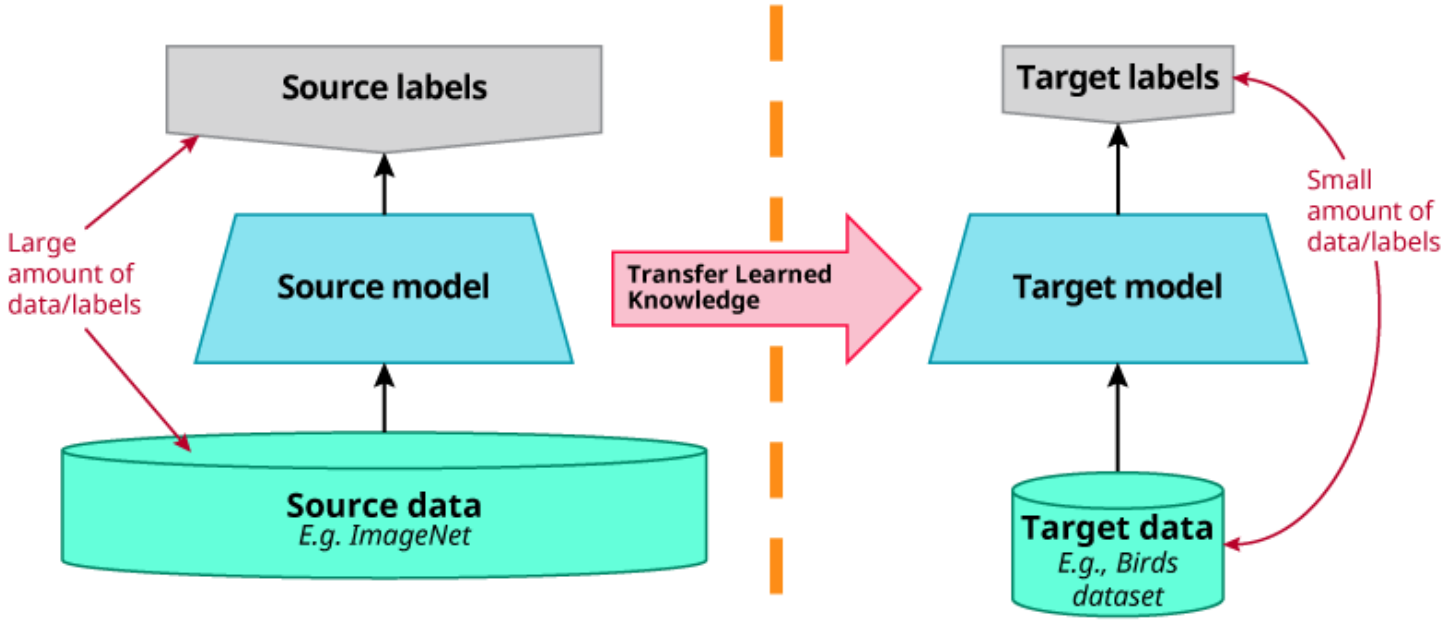
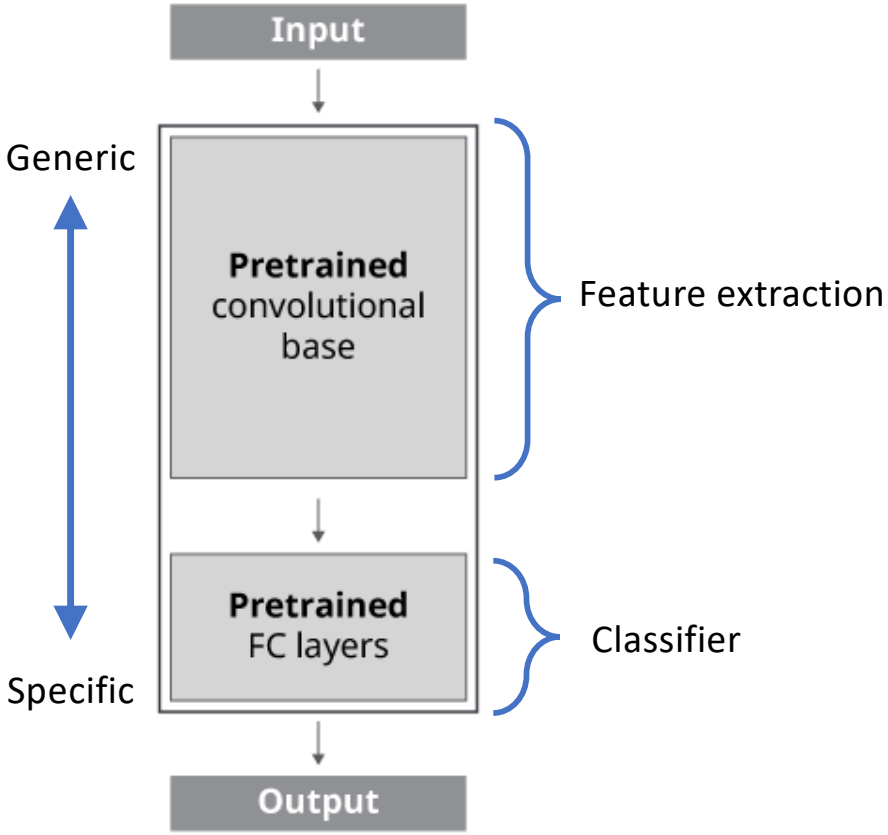


Image from: https://medium.com/the-official-integrate-ai-blog/transfer-learning-explained-7d275c1e34e2

The underlying assumption



- The **underlying assumption** of transfer learning is that generic features learned on a large enough dataset can be shared among seemingly disparate datasets.

Two ways for transfer learning with CNNs

1. Feature extraction

2. Fine-tuning

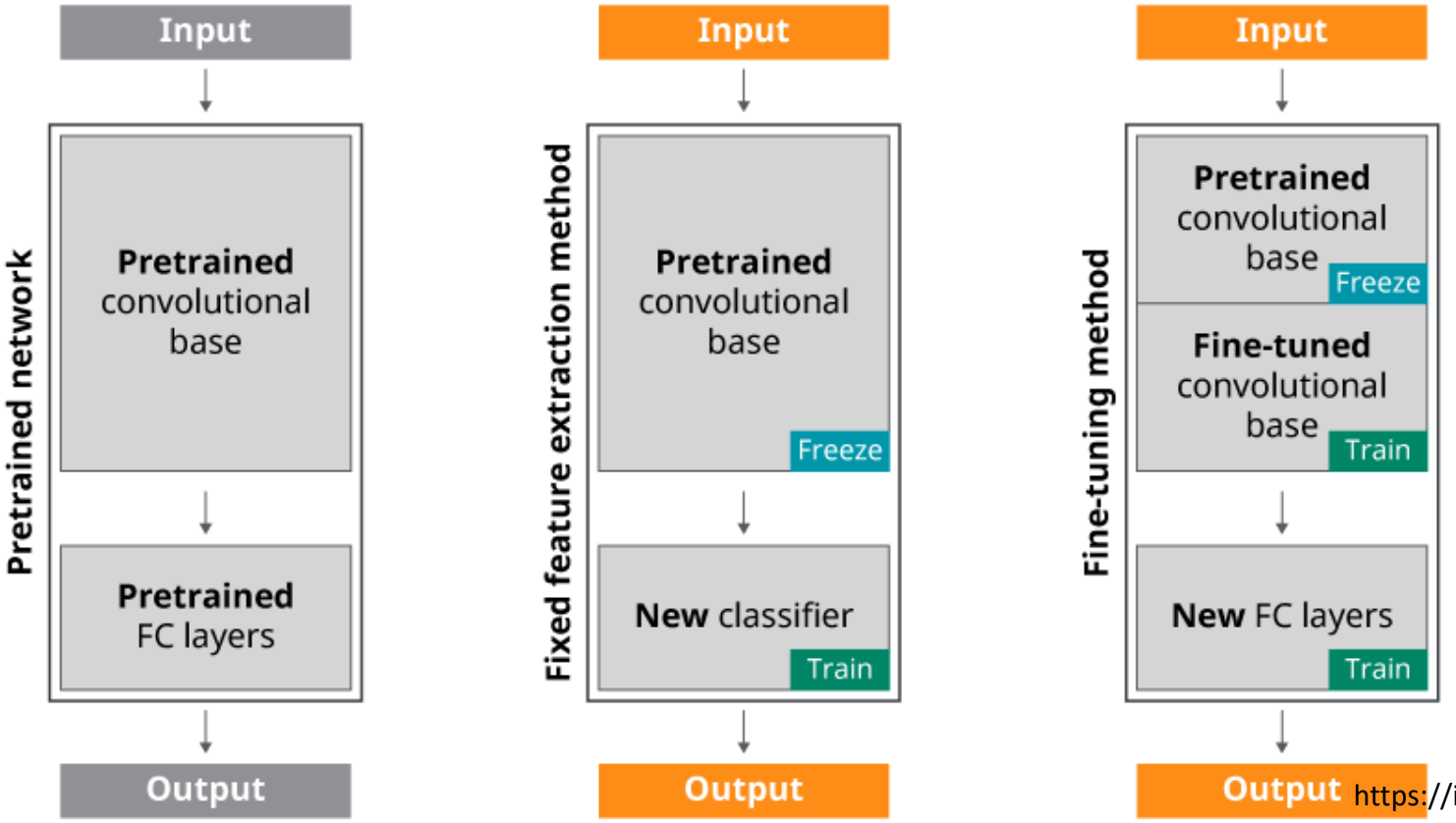
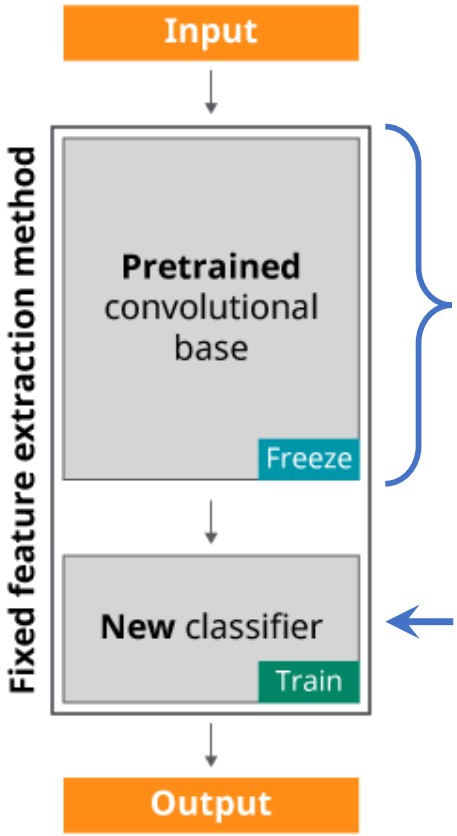


Image from:
https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

1. Feature extraction



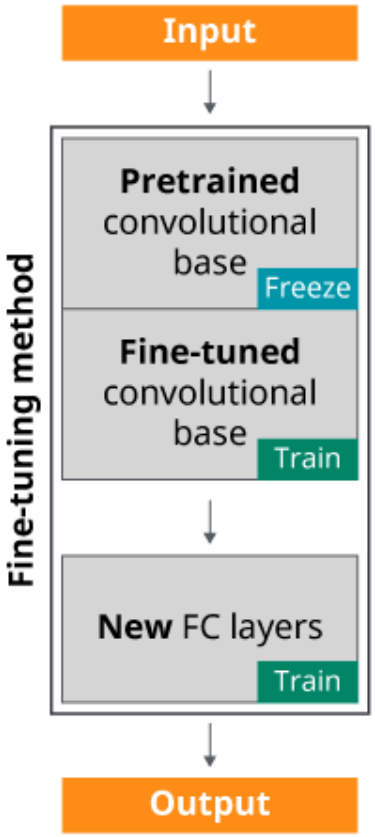
Step 1: Extract features with the pre-trained model

Step 2: train a classifier with the input features

- The new classifier could be any machine learning classifier, such as SVM, random forests and also the FC layers.

Image from:
https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

2. Fine-tuning



- **Fine-tuning**: the learnt filters are not randomly initialized, but start from the pre-trained model. These filters will be slightly adjusted.
- It is common to fine-tune only the higher layers in the convolutional base, but still freeze the early layers, since early-features appear more generic.
- Use lower learning rate while fine-tuning, 1/10 of the original learning rate would be a good start point.

Image from:
https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9

Pre-trained models

- **Q**: Do I have to pre-train a model myself on a large dataset, then use it for feature extraction or fine-tuning on a small dataset?
- **A**: Not necessary. Many models pretrained on ImageNet are open to public:
  - AlexNet,
  - VGG
  - ResNet
  - Inception
  - Xception
  - DenseNet
  - MobileNet
  - ….

Pre-trained models

- In Keras, all pre-trained models are available in `keras.application`.

```
from keras.application import VGG16

conv_base = VGG16(weights='imagenet',
                  include_top = False,
                  input_shape=(150,150,3))
```

False means not including the fully connected layers .

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201

...

All available models in Keras: https://keras.io/api/applications/

Reference for Topic 2

- Book: Francois Chollet. Deep Learning with Python. Manning. 2018.
- Stanford University course: CNNs for Visual Recognition https://cs231n.github.io/transfer-learning/
- Blog by Francois Chollet: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html
- R. Yamashita, et al. Convolutioanl neural networks: an overview and applicatin in radiology. Insights into imaging. P611-629, 2018. https://insightsimaging.springeropen.com/articles/10.1007/s13244-018-0639-9