



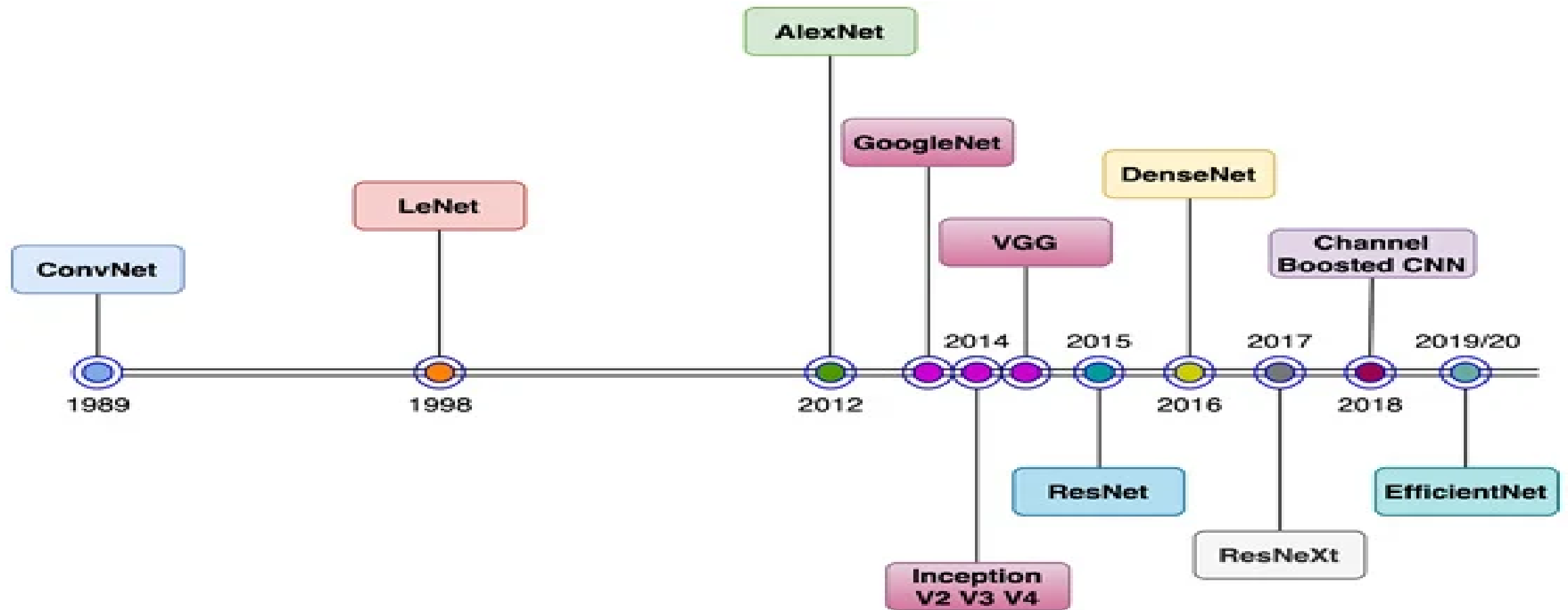
Transfer Learning and Finetuning

Presented by:

Dr. Ritesh Maurya

Associate Professor

Amity Centre for Artificial Intelligence



ImageNet Dataset

- **ImageNet** is a dataset of
 - 1.2 million labeled images for training and validation, along with 150,000 images for testing.

ImageNet Large Scale Visual Recognition Challenge (ILSVRC):

- Between 2010 -2017
- Uses ~1000 categories, each with ~1000 images

Object detection

Easiest classes

butterfly (93)



bird (78)



dog (84)



tiger (77)



Hardest classes

lamp (15)



ski (12)



flute (15)



microphone (11)



<https://image-net.org/index.php>



VGGNet



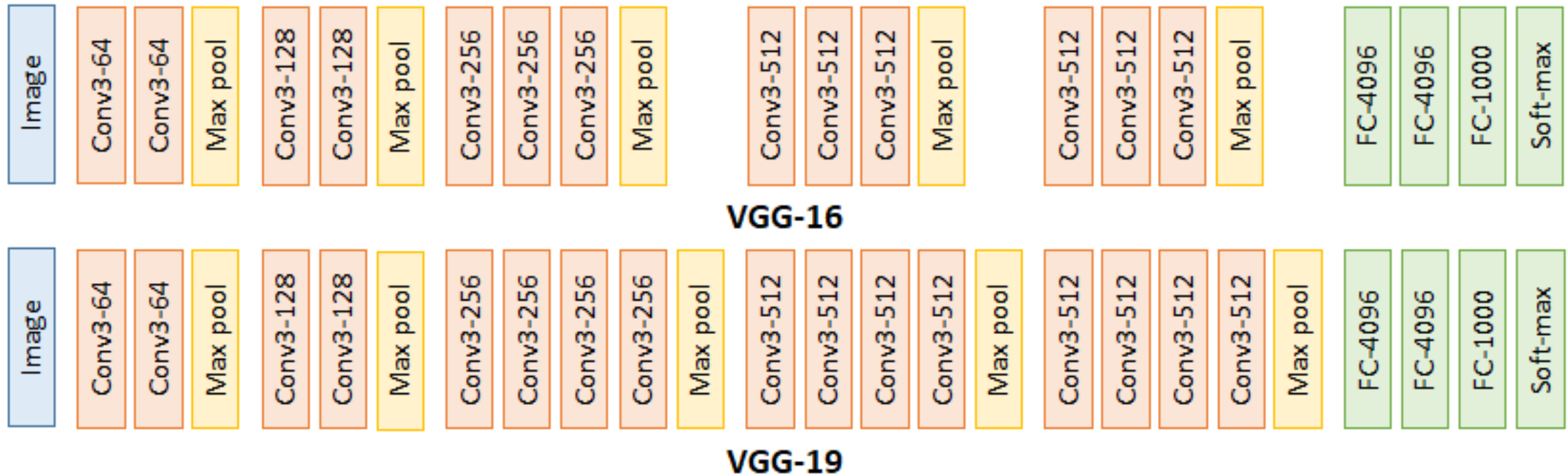


VGGNet

- Developed by Visual Geometry Group in 2014
- VGG16 was 2nd in ILSVRC(ImageNet Large Scale Visual Recognition Challenge) challenge 2014 (top-5 classification error of 7.32%)
- Characterized by **Simplicity and Depth**
 - All **Conv layers** with 3x3 filters and stride 1, SAME padding
 - All **max polling** layers 2x2 filters, stride 2
- **VGG16**: 16-layer CNN (16 layers with trainable parameters, over 134 million parameters); **VGG19**: 19-layer CNN (more than 144 million parameters)

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

VGGNet



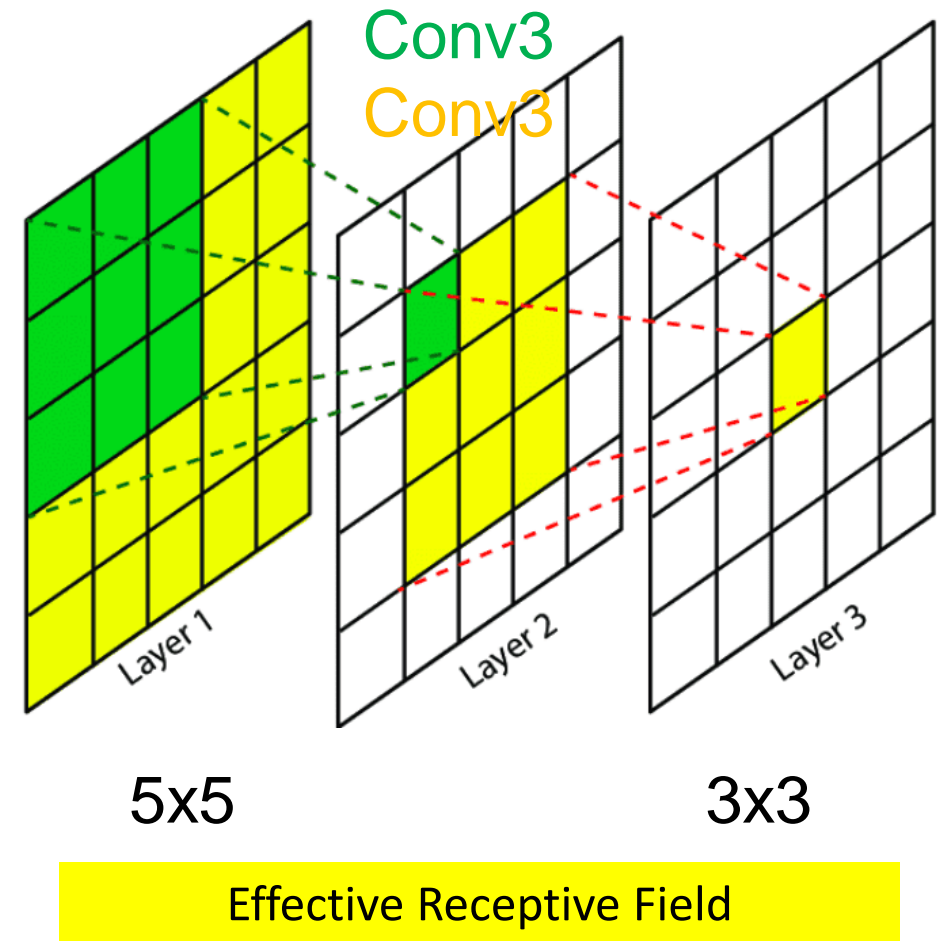
Conv= 3x3 filter, s=1, same
Max pool= 2x2, s=2 (5 Max pooling layers)

ReLU activation in all hidden units
Softmax activation in output units



Stacking of Conv

- Multiple Stacked Conv layers lead to Wide Receptive Field
- In VGG, varying filter sizes are implemented by stacking Conv layers with fixed filter sizes






GoogleNet

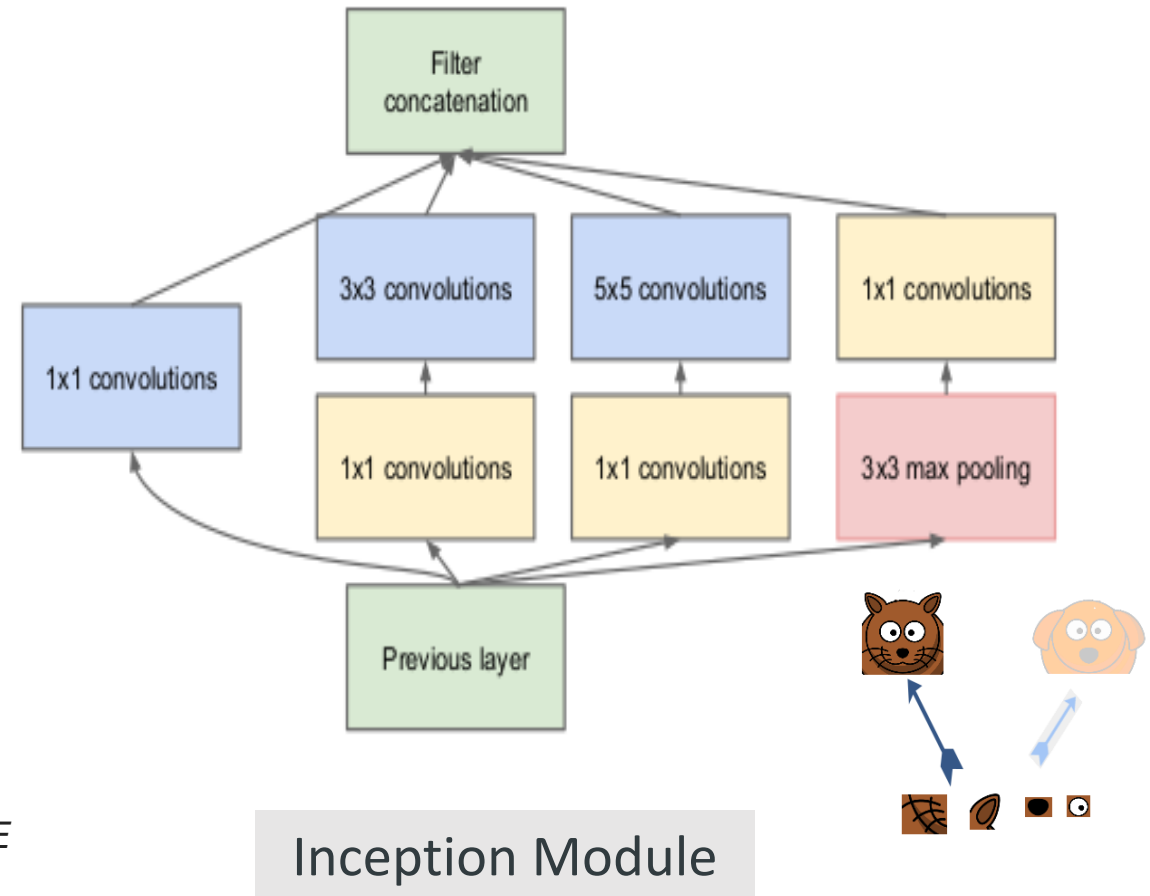




GoogLeNet (Inception v1)

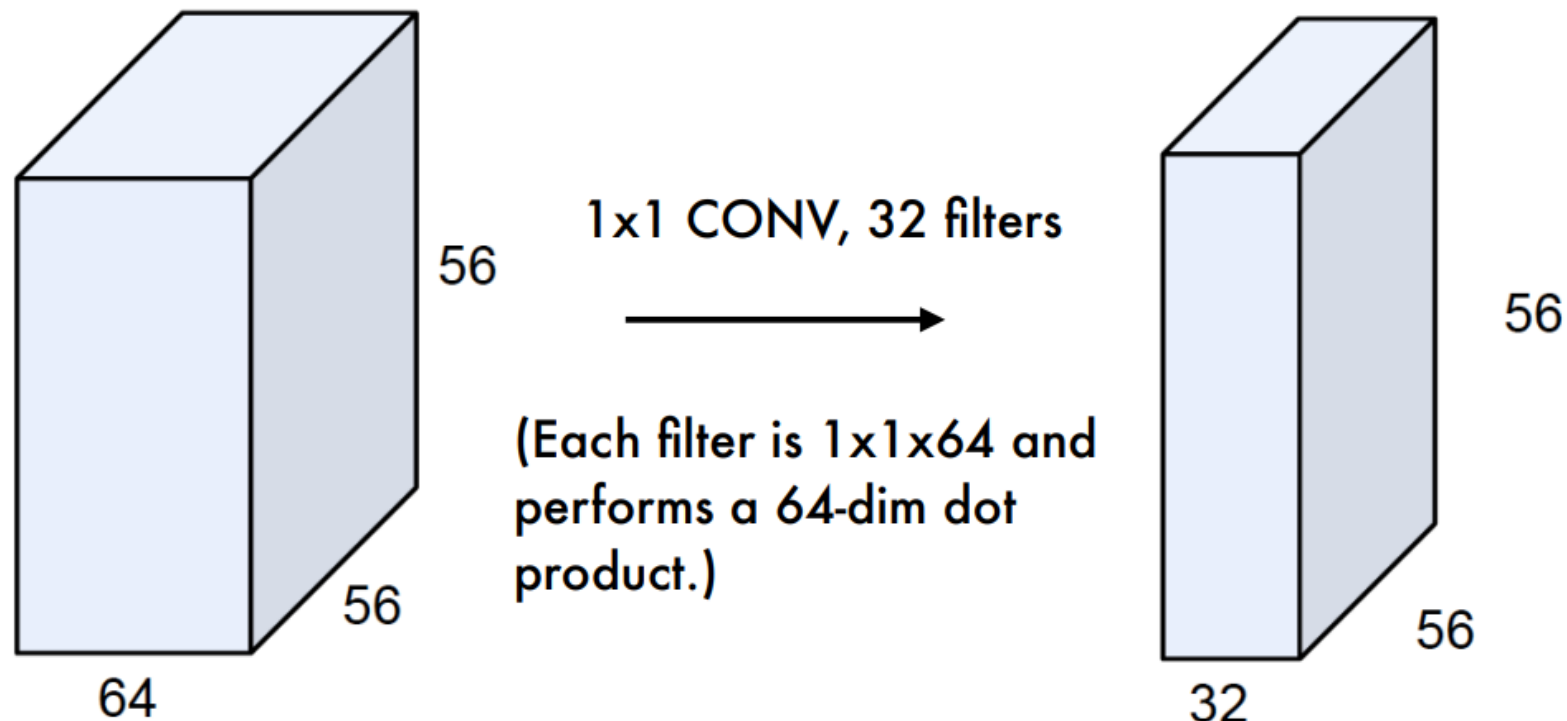
- Developed by Google in 2014
- 1st position in ILSVRC challenge 2014 (top-5 classification error of 6.66%)
- 22-layers with trainable parameters (27 layers including Max Pool layers)
- Parameters: 5 million (V1), 23 million (V3)
- Contains Inception Modules 

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1-9. 2015.



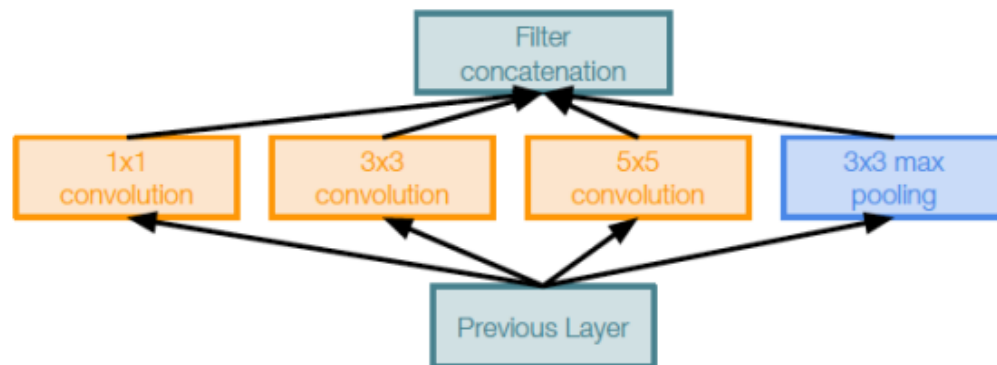


1x1 convolutions for projection

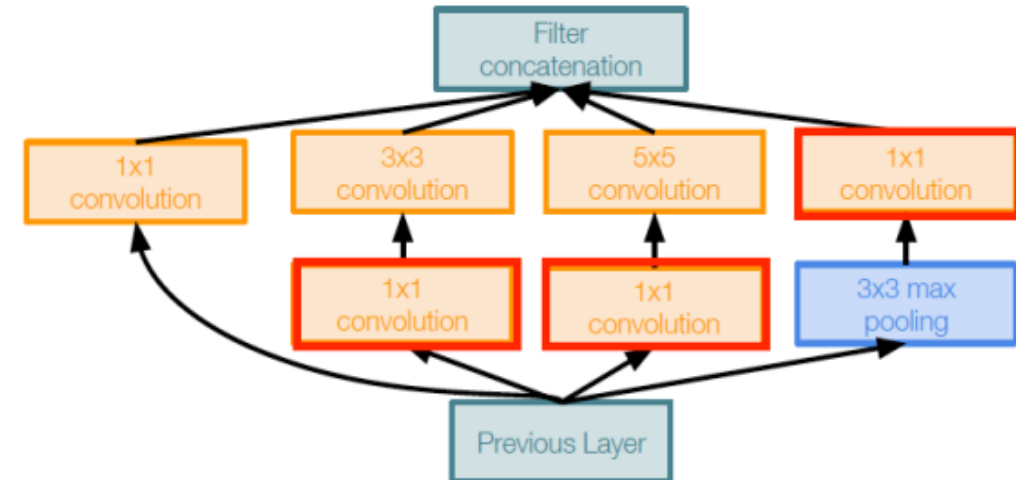


- Preserves spatial dims but reduces depth!
- Projects depth to lower dimension (essentially combining activation maps)

Inception module with dimensionality reduction



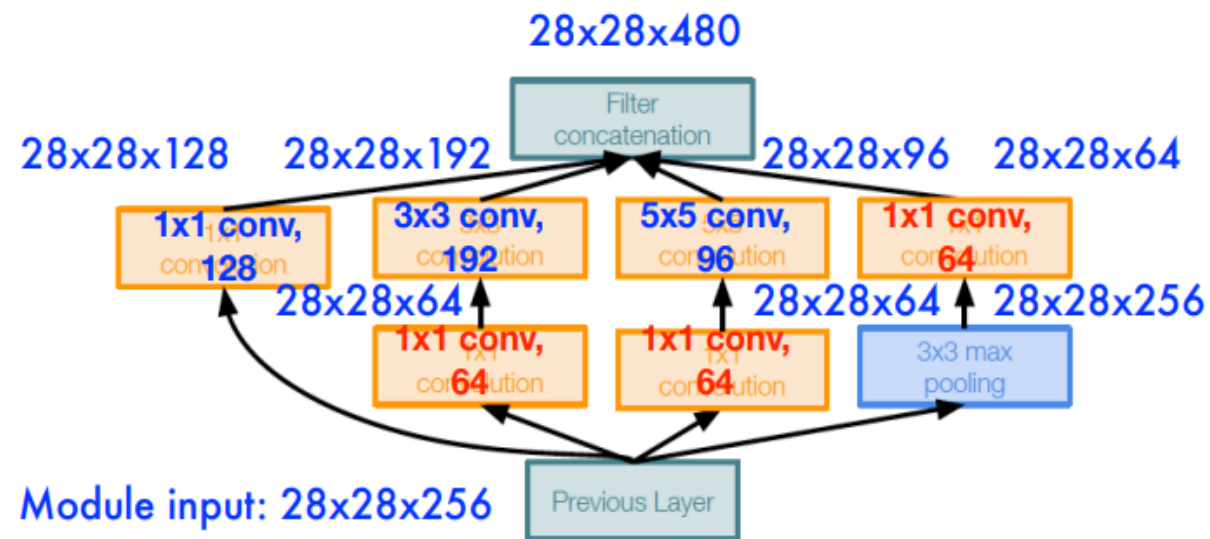
Naive Inception module



With dimensionality reduction



Inception module with dimensionality reduction



With dimensionality reduction

Same design, but with '1x1 conv, 64 filter bottlenecks:

Conv Ops:

[1x1 conv, 64] 28x28x64x1x1x256
 [1x1 conv, 64] 28x28x64x1x1x256
 [1x1 conv, 128] 28x28x128x1x1x256
 [3x3 conv, 192] 28x28x192x3x3x64
 [5x5 conv, 96] 28x28x96x5x5x64
 [1x1 conv, 64] 28x28x64x1x1x256

Total of 358M ops (vs 854M)!

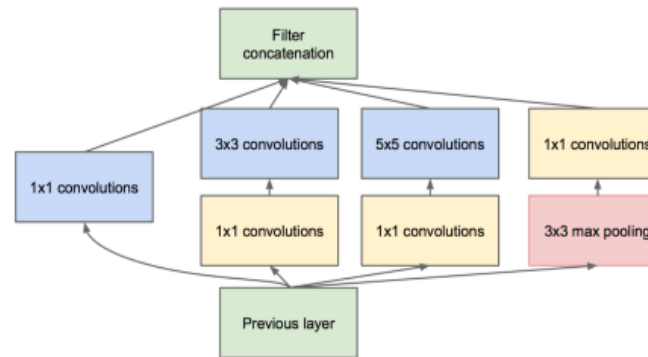
Note that bottleneck also reduces depth after pooling layer.



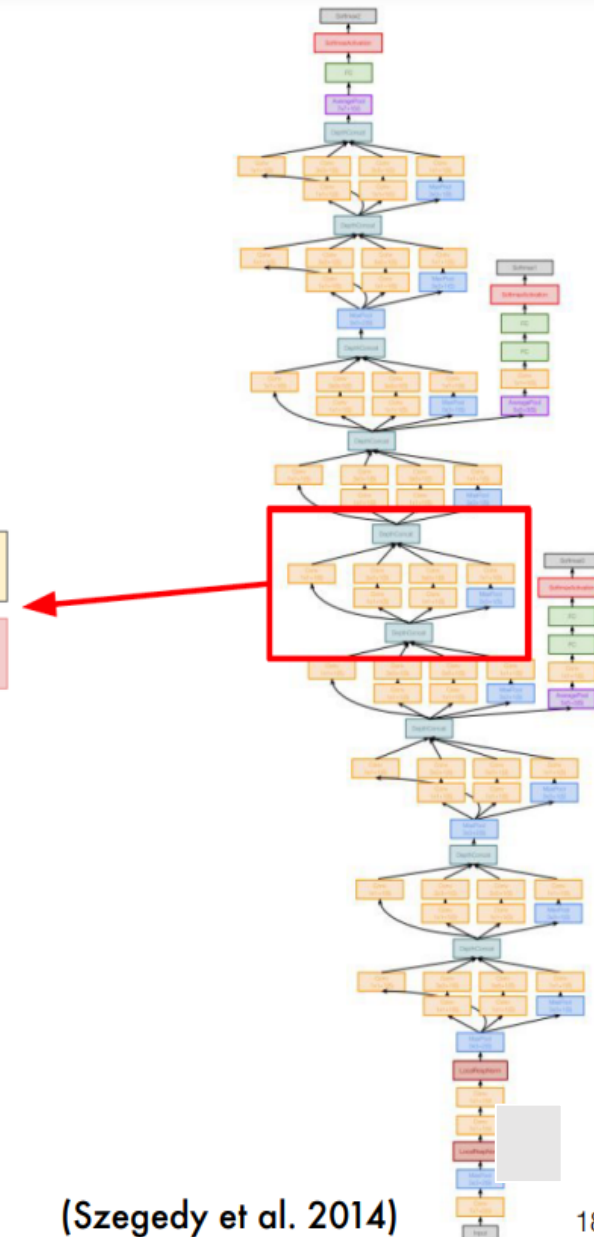
GoogLeNet

**Deeper networks,
with computational efficiency**

- 22 layers
- Efficient 'inception' module
- No FC layers!
- Only 5 million parameters!
(1/12th of AlexNet)
- ILSVRC'14 classification winner
(6.7% top-5 error)



Inception module



(Szegedy et al. 2014)



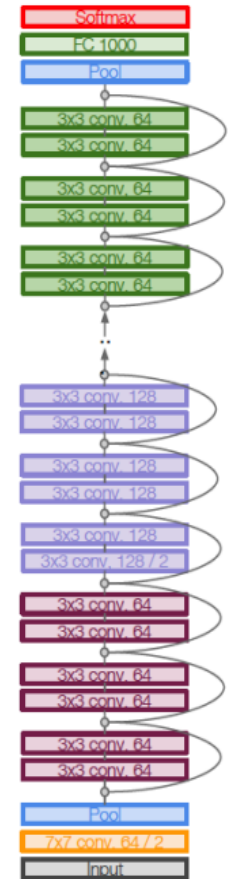
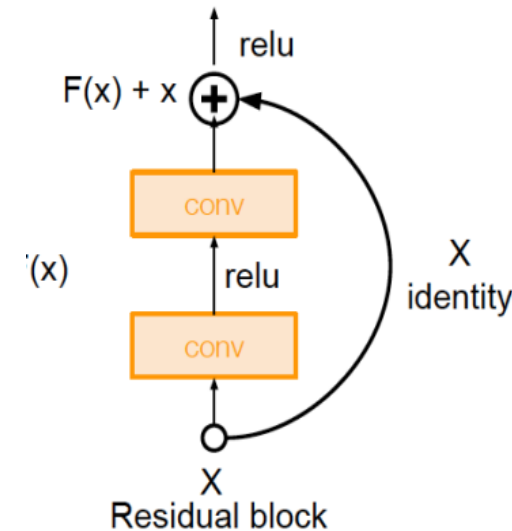
ResNet





Residual Block

- **Skip connection** skips training from a few layers and connects directly to the output
- Instead of learning the underlying mapping $H(x)$ from stacked layers, let network learn the **residual** $F(x) = H(x) - x$
- Hence, after **adding identity**, $F(x) + x = H(x)$
- Speeds learning by reducing the impact of vanishing gradients, avoid degradation
- Enable development of Deeper Networks



(He et al. 2015)



ResNet

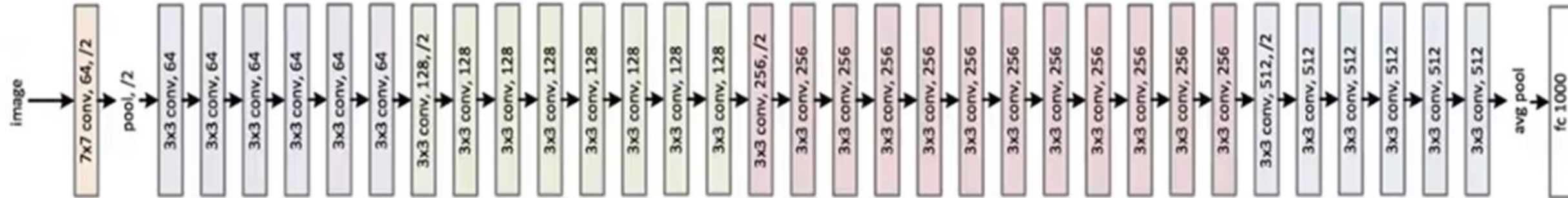
- Proposed by Shaoqing Ren, Kaiming He, Jian Sun, and Xiangyu Zhang
- Vanishing Gradient problem of Deep NN: with increase in depth
- 1st position in ILSVRC challenge 2015 (top-5 classification error of 3.57%)
- ResNet-50(50 conv layers) parameter count of approximately 25.6 million makes it a moderately large network compared to earlier architectures.

He, Kaiming, 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, DOI: <https://arxiv.org/abs/1512.03385>.



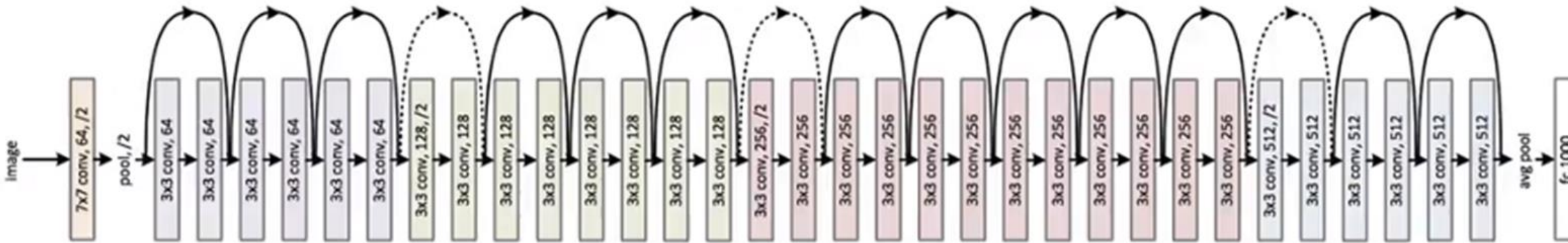
Plain

34-layer plain



ResNet

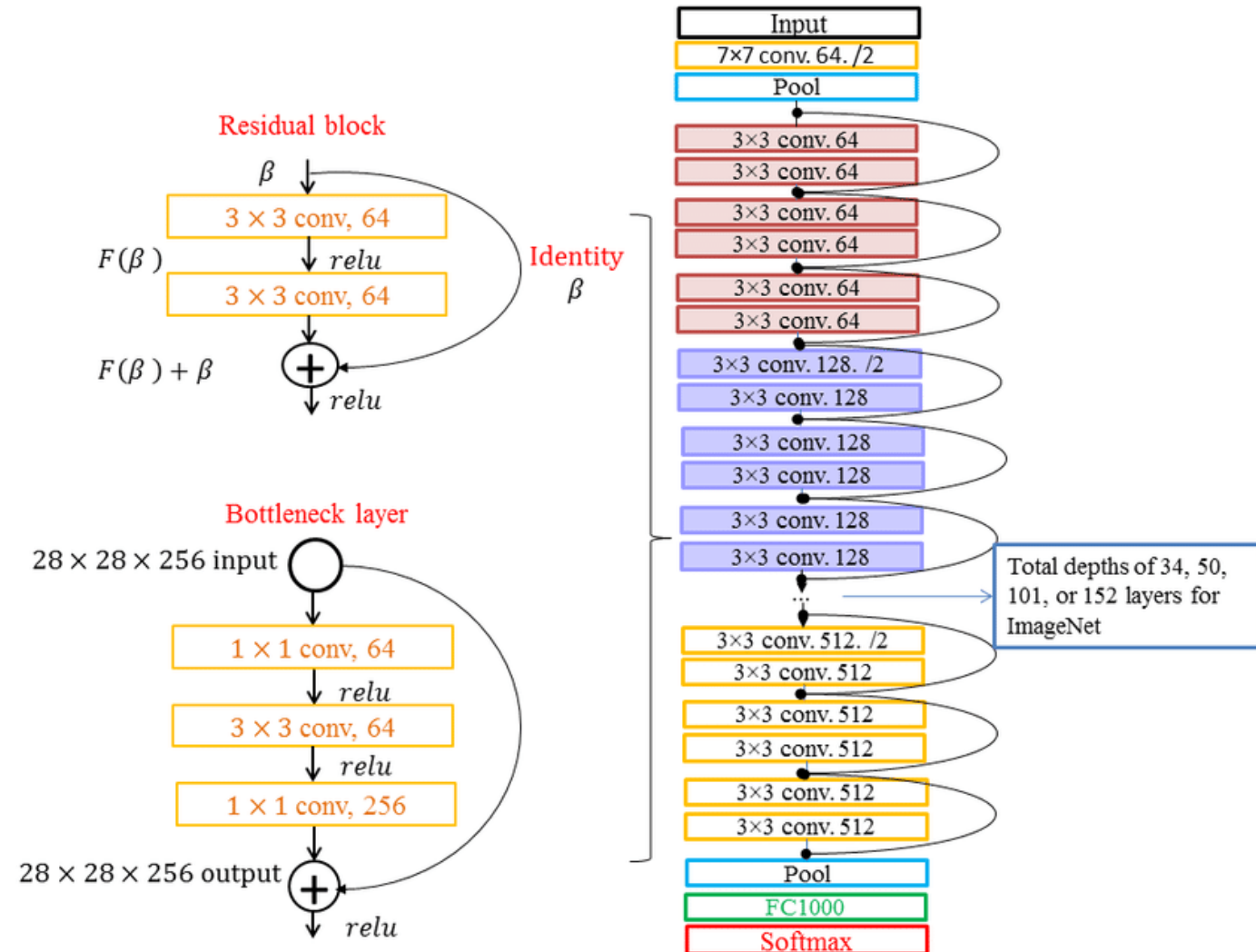
34-layer residual



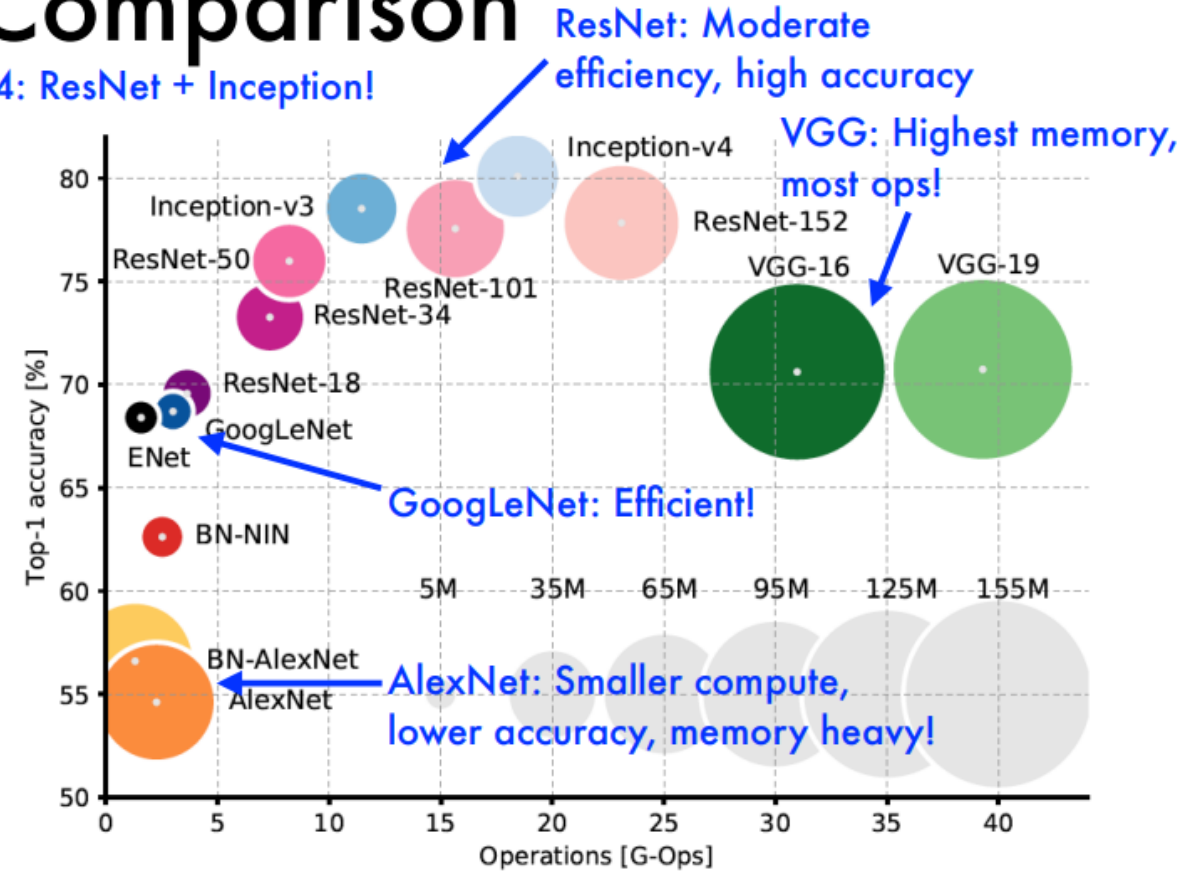
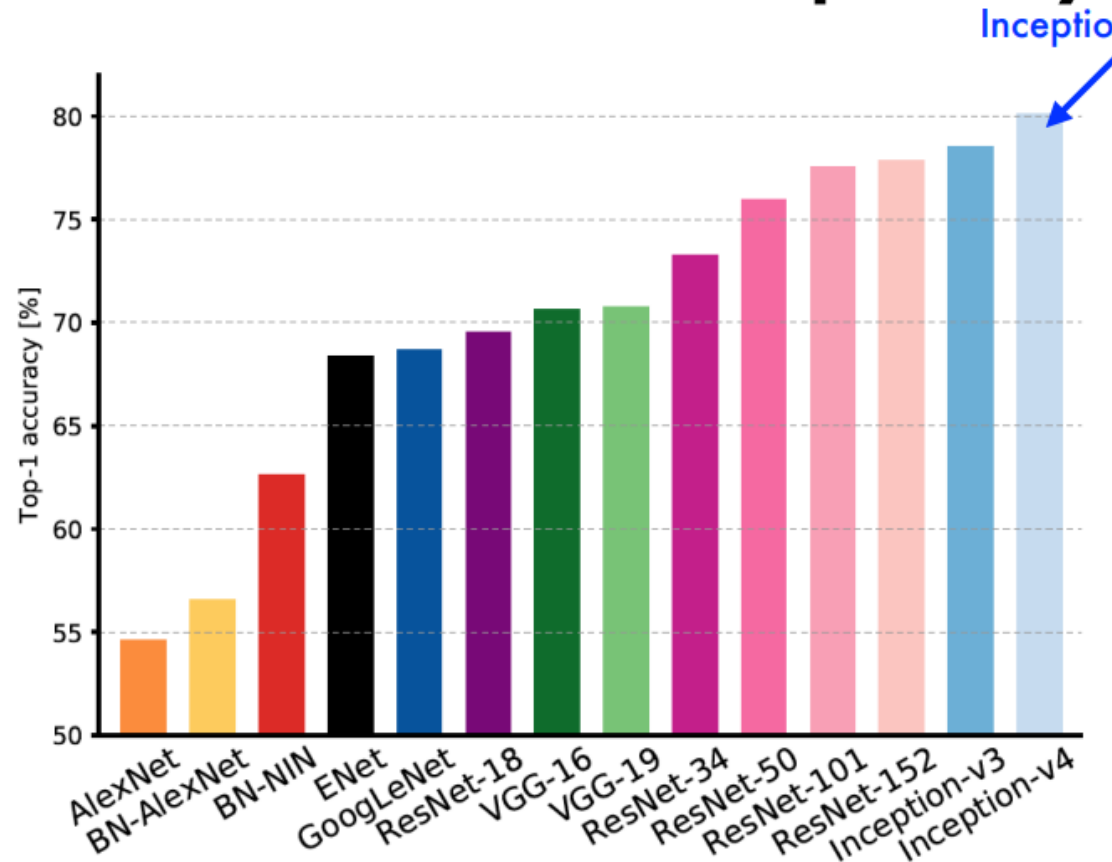


What's Novel in ResNet?

Residual Connection
High Accuracy
Bottleneck layers



Complexity Comparison



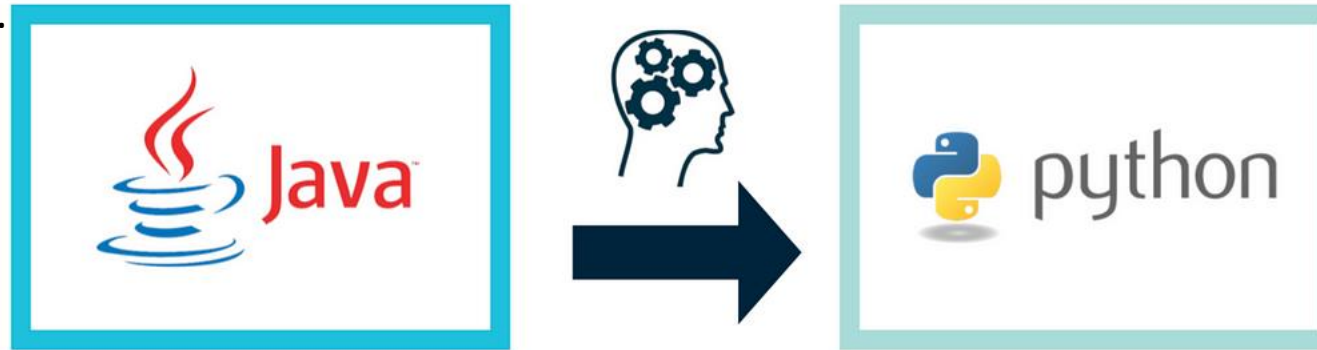
An Analysis of Deep Neural Network Models for Practical Applications (Canziani et al. 2017)



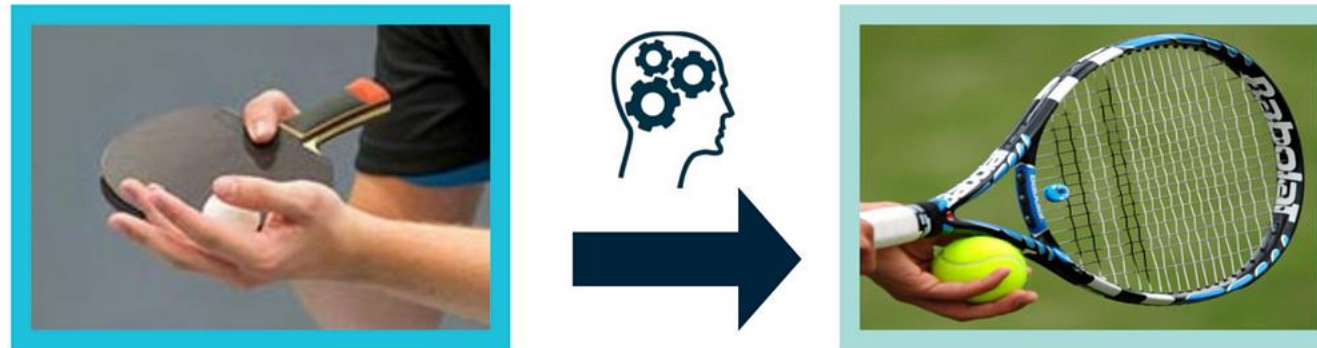
Concept of Transfer Learning



Transfer Learning



a) Transferring Learned knowledge from Java to Python.



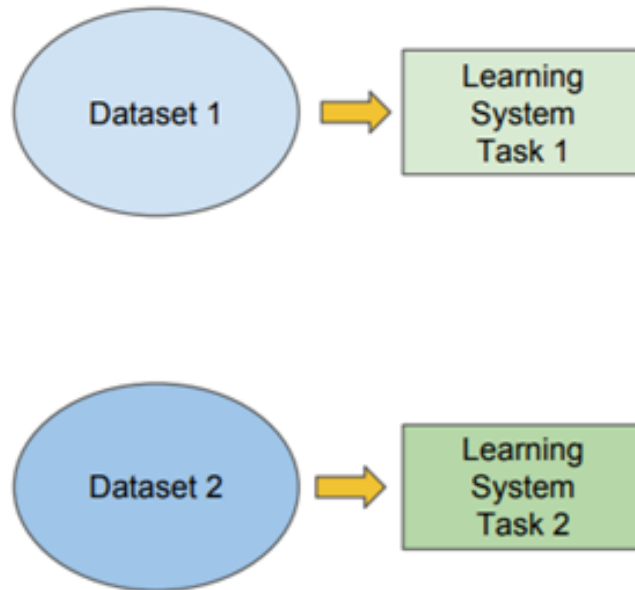
b) Transferring Learned knowledge Table Tennis to Tennis.

Traditional ML

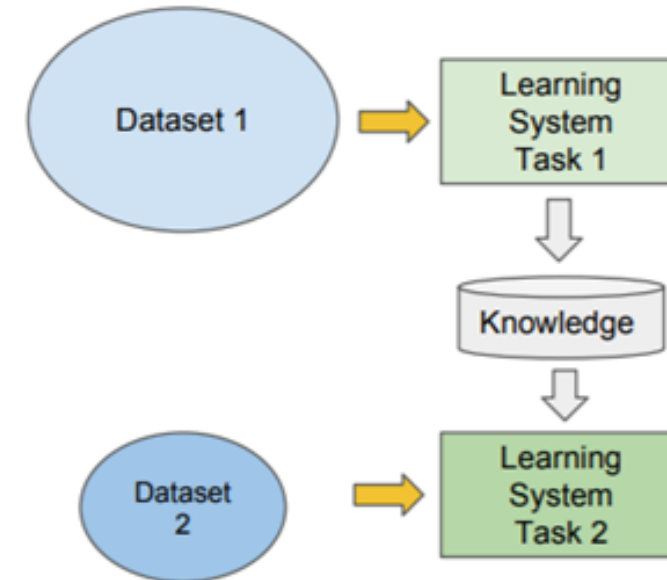
vs

Transfer Learning

- Isolated, single task learning:
 - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



- Learning of a new tasks relies on the previous learned tasks:
 - Learning process can be faster, more accurate and/or need less training data

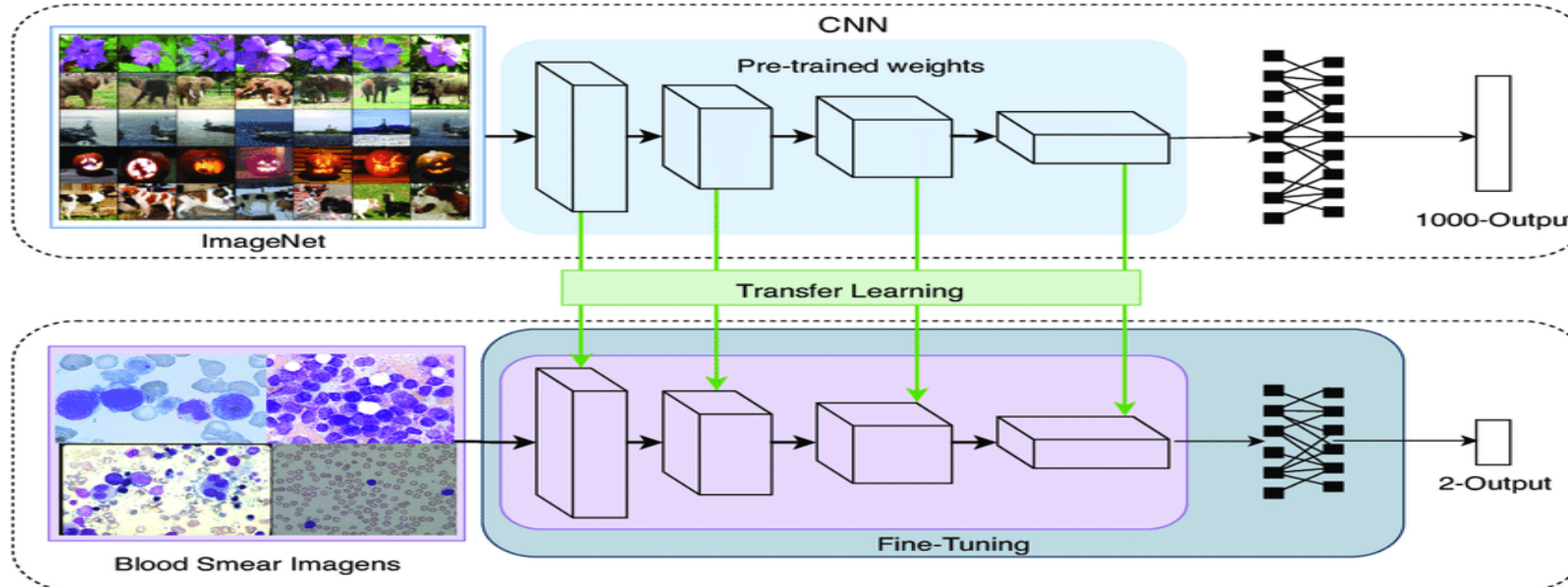




Transfer Learning

- Improvement of learning in a **new** task through the *transfer of knowledge* from a **related** task that has already been learned.
- Weight initialization for CNN
- Strategy
 - Fine-tuning the pre-trained deep CNNs.

When to finetune your model?



- New dataset is small and similar to the original dataset
 - fine-tune through the some of the last layers
- New dataset is large and very different from the original dataset
 - fine-tune through the some or entire network



Freeze or fine-tune?

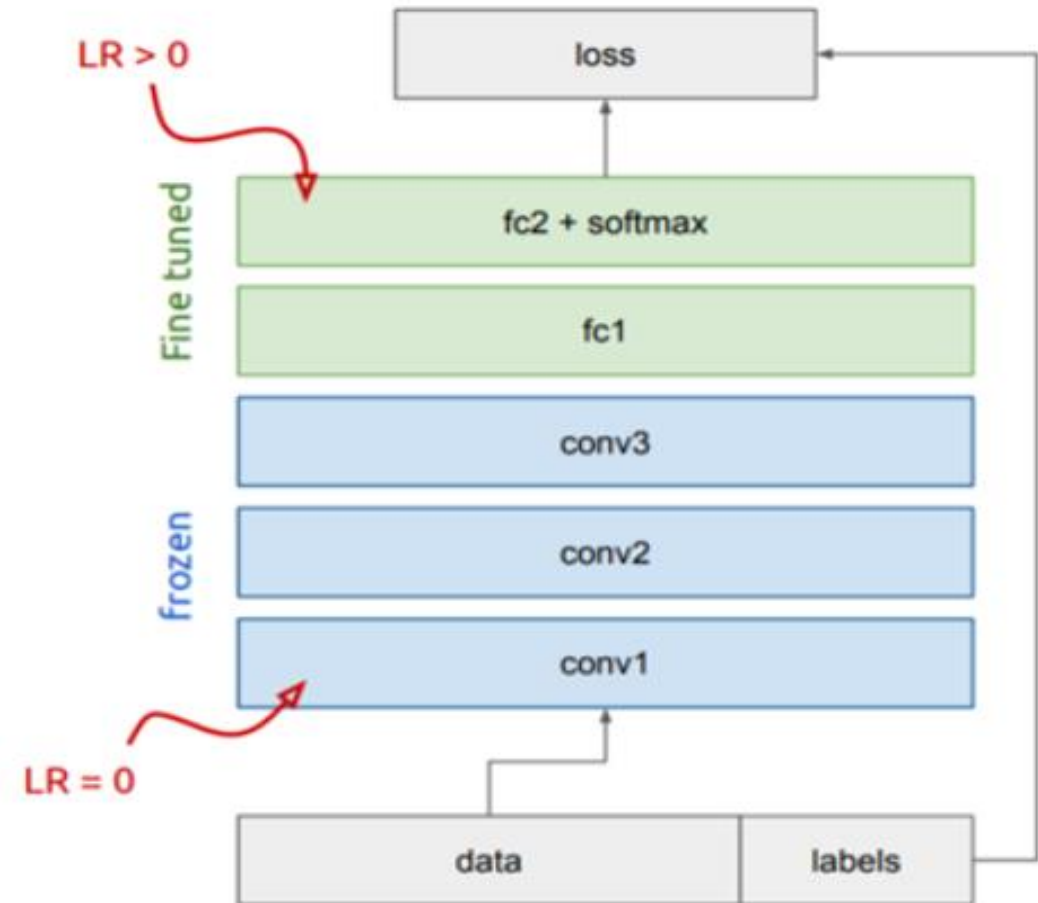
Bottom n layers can be frozen or fine tuned.

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

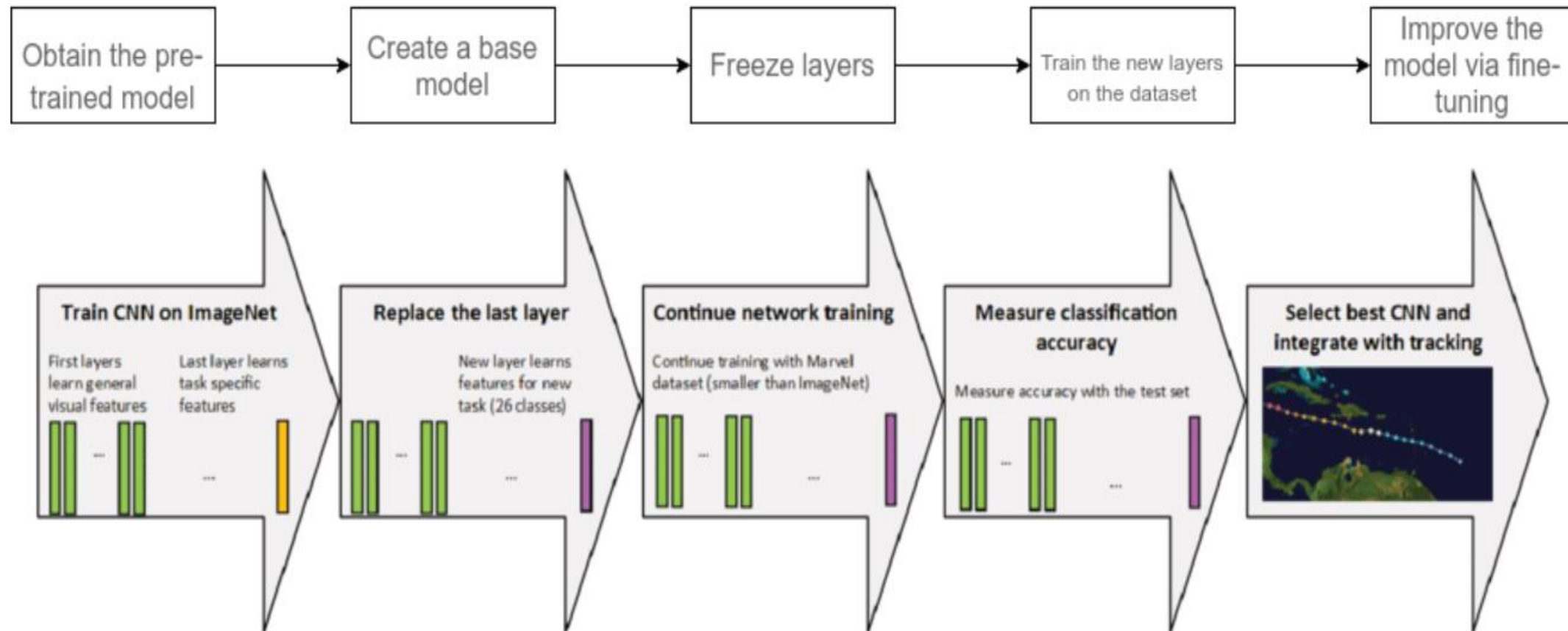
Which to do depends on target task:

- **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



Steps in Transfer Learning



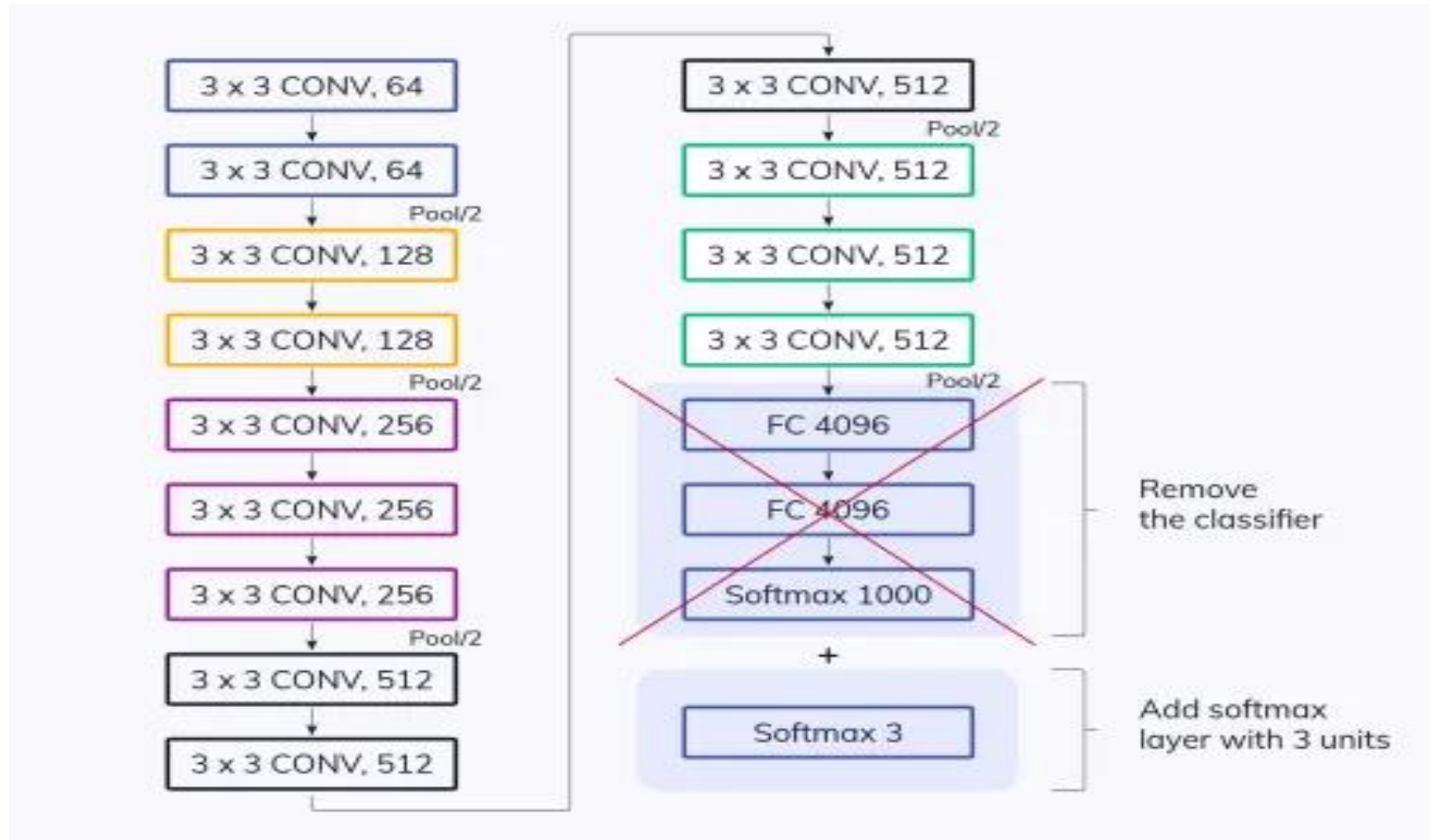


1-Obtain pre-trained model

- VGG-16
- VGG-19
- Inception V3
- Xception
- ResNet-50



2. Create a base model

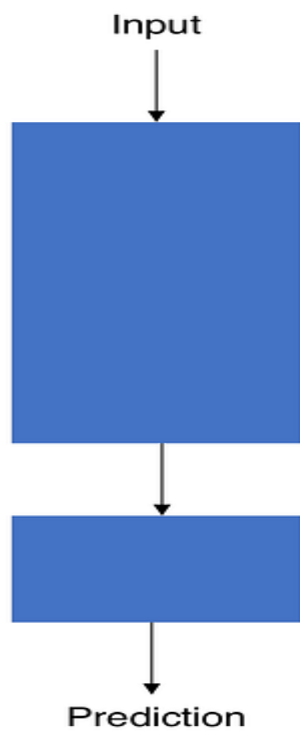




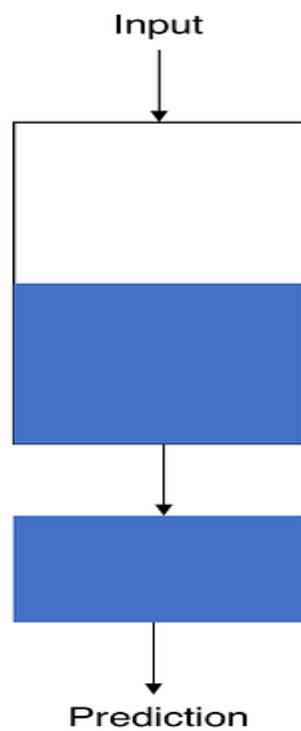
3. Freeze layers

- Freezing the starting layers from the pre-trained model is essential to avoid the additional work of making the model learn the basic features.
- If we do not freeze the initial layers, we will lose all the learning that has already taken place. This will be no different from training the model from scratch and will be a loss of time, resources, etc.

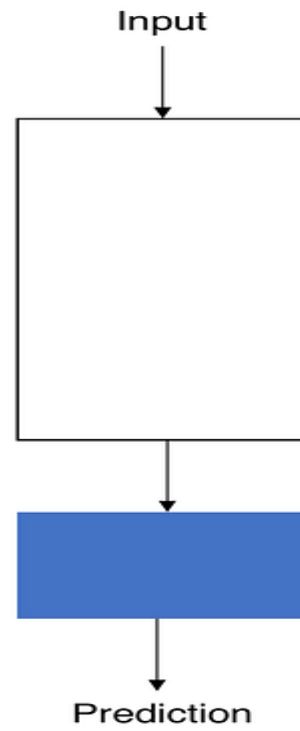
Strategy 1
Train the
entire model



Strategy 2
Train some layers and
leave the others frozen



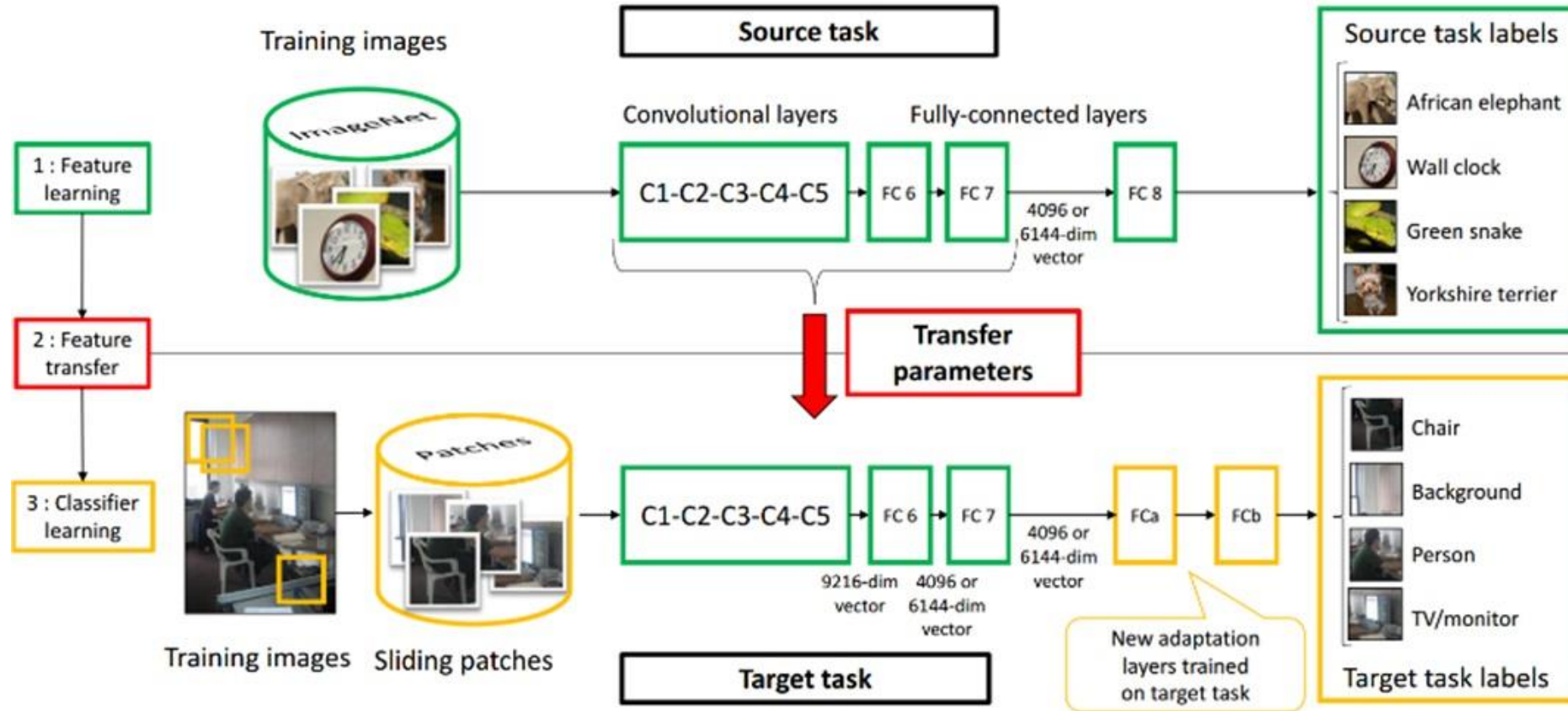
Strategy 3
Freeze the
convolutional base



Legend:



4. Add new trainable layers



- The only knowledge we are reusing from the base model is the feature extraction layers. We need to add additional layers on top of them to predict the specialized tasks of the model. These are generally the final output layers.



5. Train the new layers

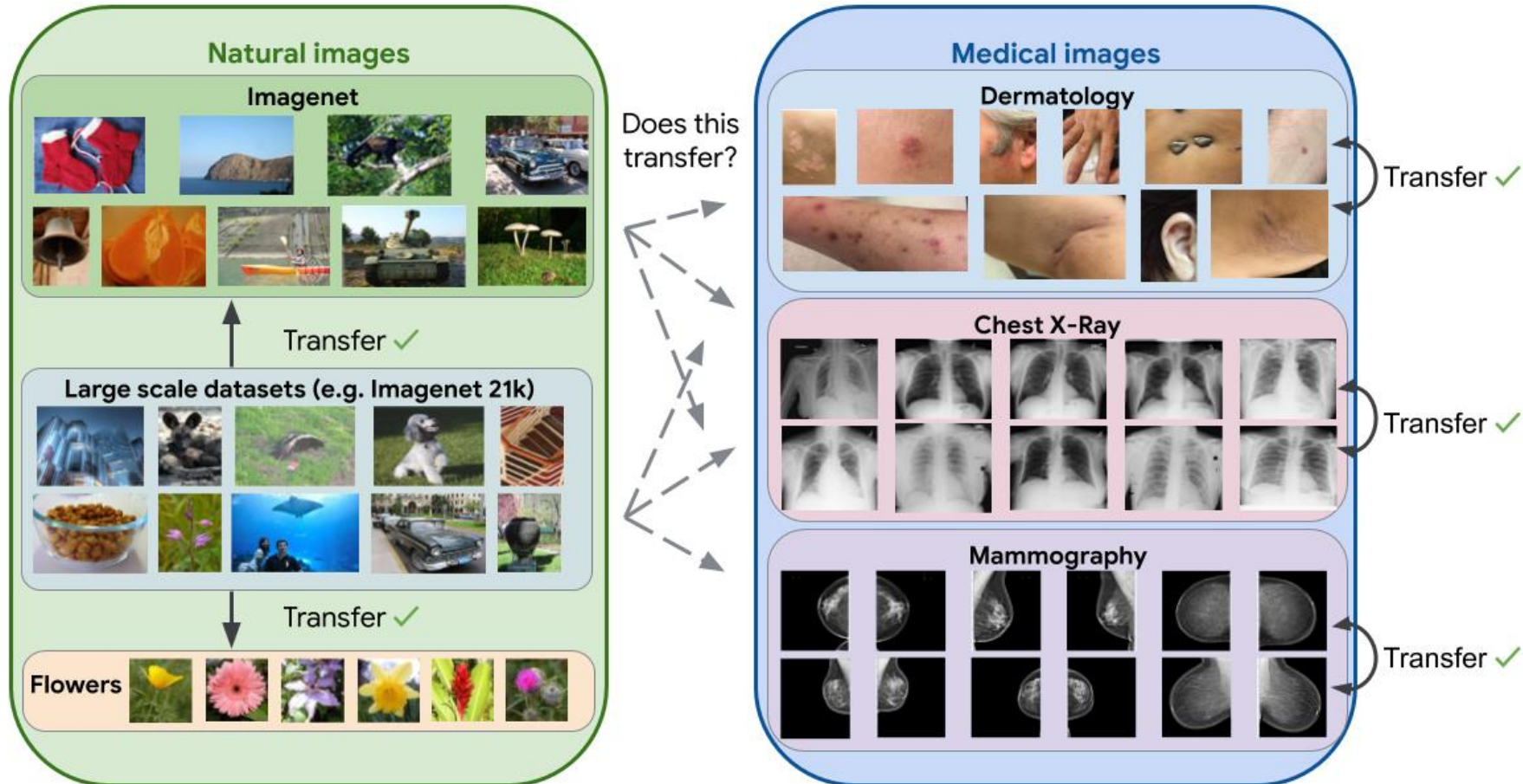
- The pre-trained model's final output will most likely differ from the output we want for our model. For example, pre-trained models trained on the ImageNet dataset will output 1000 classes.
- However, we need our model to work for two classes. In this case, we have to train the model with a new output layer in place.



6. Fine-tune your model

- One method of improving the performance is fine-tuning.
- Fine-tuning involves unfreezing some part of the base model and training the entire model again on the whole dataset at a very low learning rate. The low learning rate will increase the performance of the model on the new dataset while preventing overfitting.

Application of Transfer Learning



Thank You.