High Impact Skills Development Program

in Artificial Intelligence, Data Science, and Blockchain

Module: [Computer Vision]

Lecture 4: [Transfer Learning and Pre-Trained ConvNets]

Instructor: [Dr. Rabia Irfan]
[Assistant Professor], SEECS, NUST

Agenda

- Popular CNN Architectures
- Transfer Learning
    - Feature-Extractors
    - Fine Tuning

Popular CNN Architectures

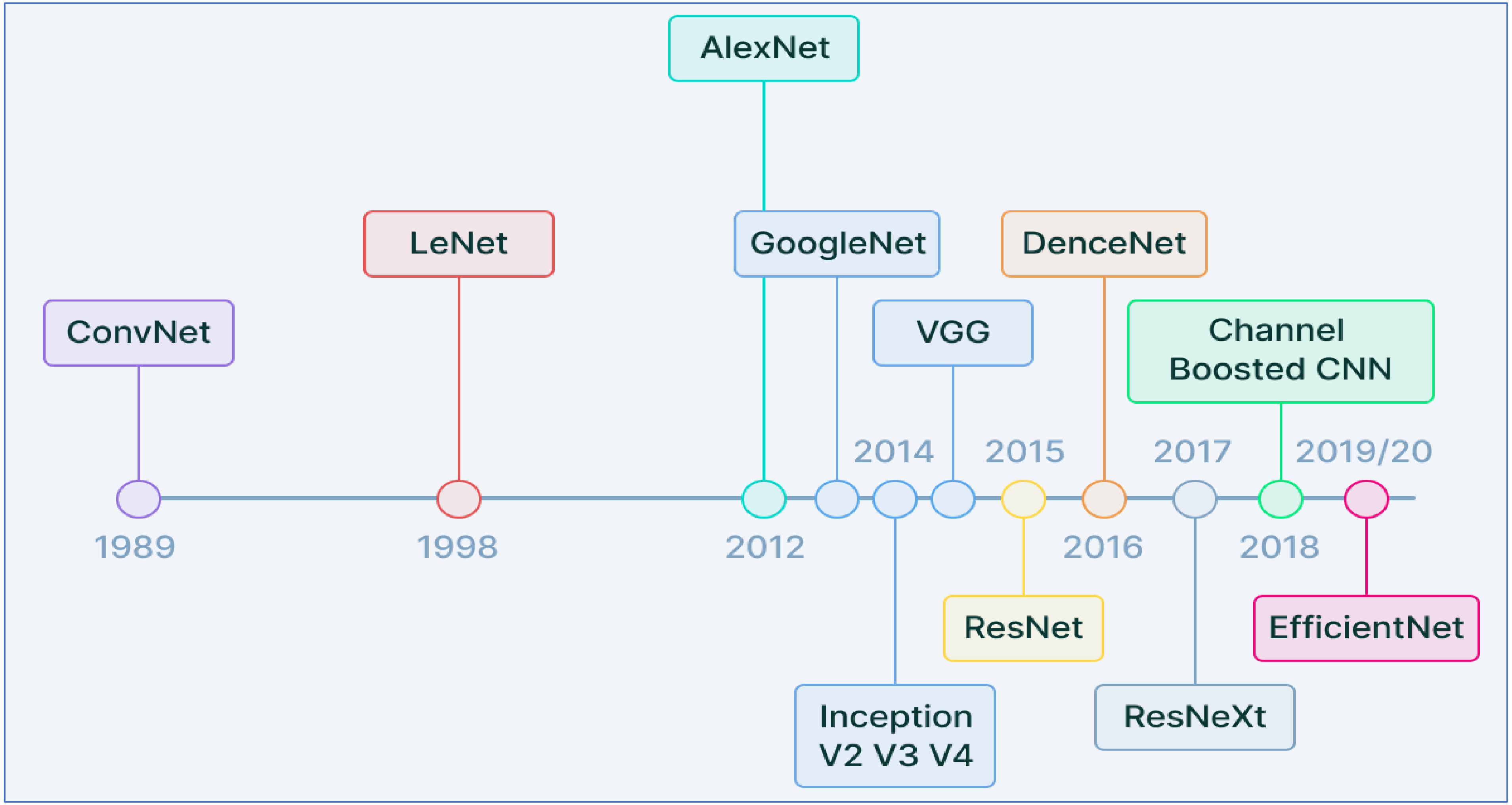
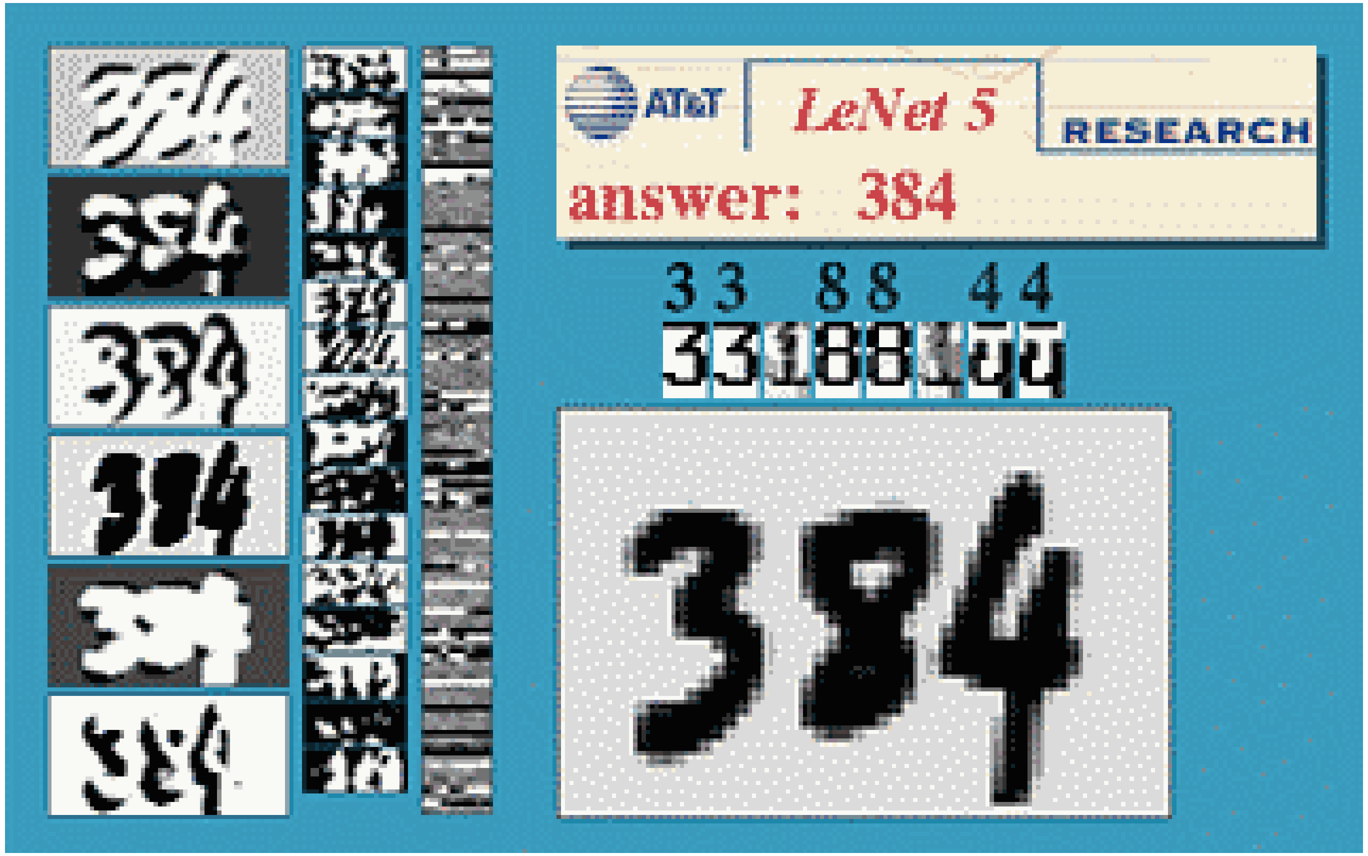


Image Credit: https://www.v7labs.com/blog/convolutional-neural-networks-guide

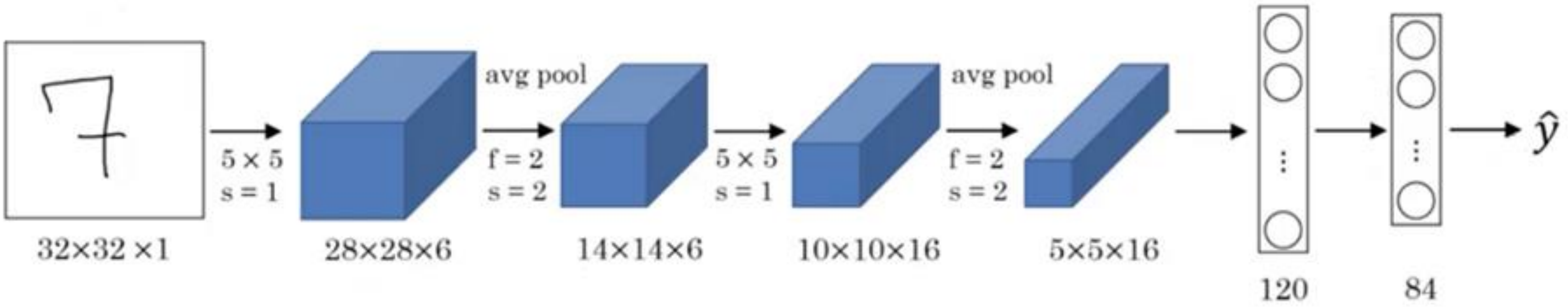
LeNet-5

- Yann Lecun's LeNet-5 model was developed in **<u>1998</u>** to identify **<u>handwritten digits for zip code recognition</u>** in the postal service.



LeNet-5

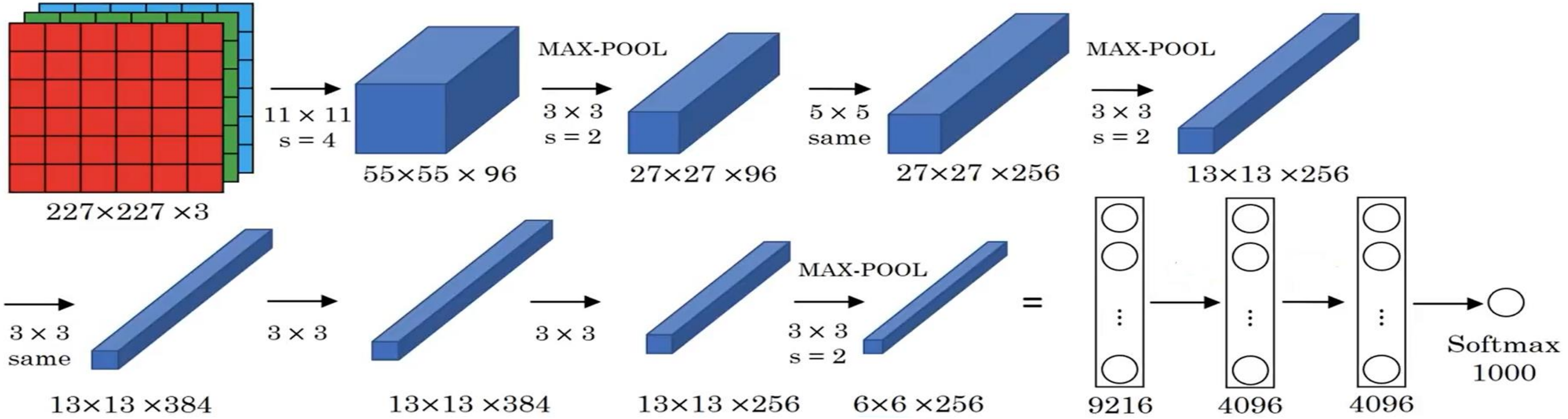
- It was trained on 2D **grayscale images** with a size of **32*32*1**.
- The subsampling layers use a form of average pooling.
- Parameters Learned: **60,000**



[Paper: Gradient-based learning applied to document recognition]

AlexNet

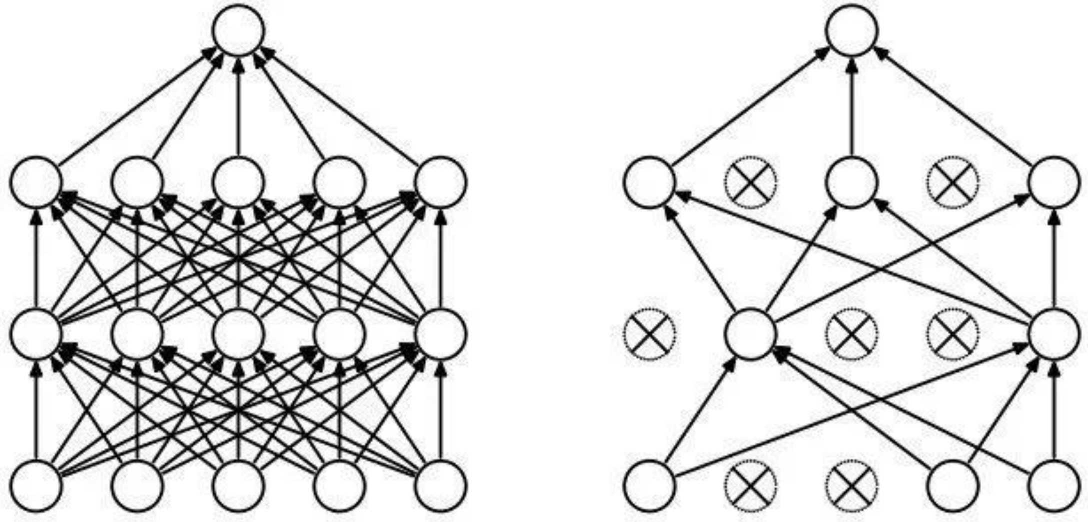
- AlexNet was developed by **Alex Krizhevsky** et al. in **2012** to compete in the **ImageNet competition.**
- It starts with 227 x 227 x 3 images and the next convolution layer applies 96 of 11 x 11 filter with stride of 4.
- It had **five** convolutional-pooling layer blocks followed by **three** fully connected dense layers for classification.



AlexNet

- AlexNet was trained on the **Imagenet** dataset with 15 million high-resolution images.
- **ReLU activation** function was used between convolution layers and pooling layers for the **first time** as well as the overlapping pooling with stride < window size.
- Another problem that this architecture solved was reducing the **over-fitting** by using a Dropout layer after every FC layer.
- Parameters Learned: 60 Million

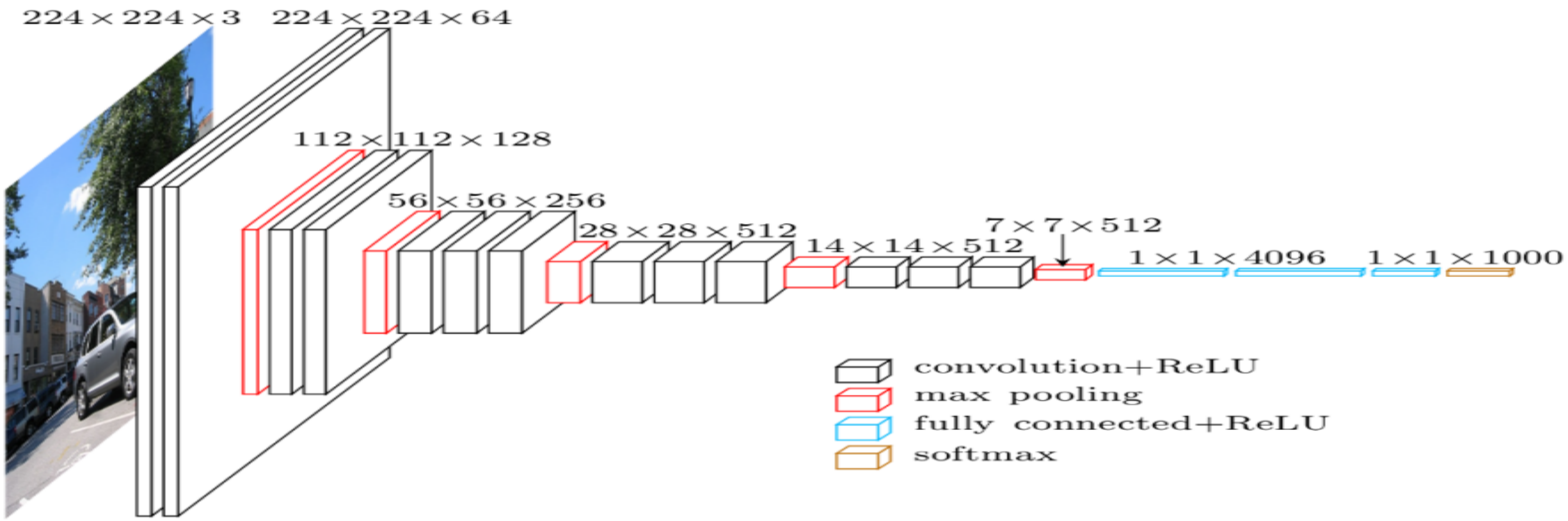
Imagenet classification with deep convolutional neural networks
A Krizhevsky, I Sutskever... - Advances in neural ..., 2012 - proceedings.neurips.cc
We trained a large, deep convolutional neural network to classify the 1.3 million high-resolution images in the LSVRC-2010 ImageNet training set into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 39.7\% and 18.9\% which is considerably better than the previous state-of-the-art results. The neural network, which has 60 million parameters and 500,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and two globally connected layers with a final ...
Save Cite Cited by 118761 Related articles All 111 versions



(a) Standard Neural Net (b) After applying dropout.

VGGNet

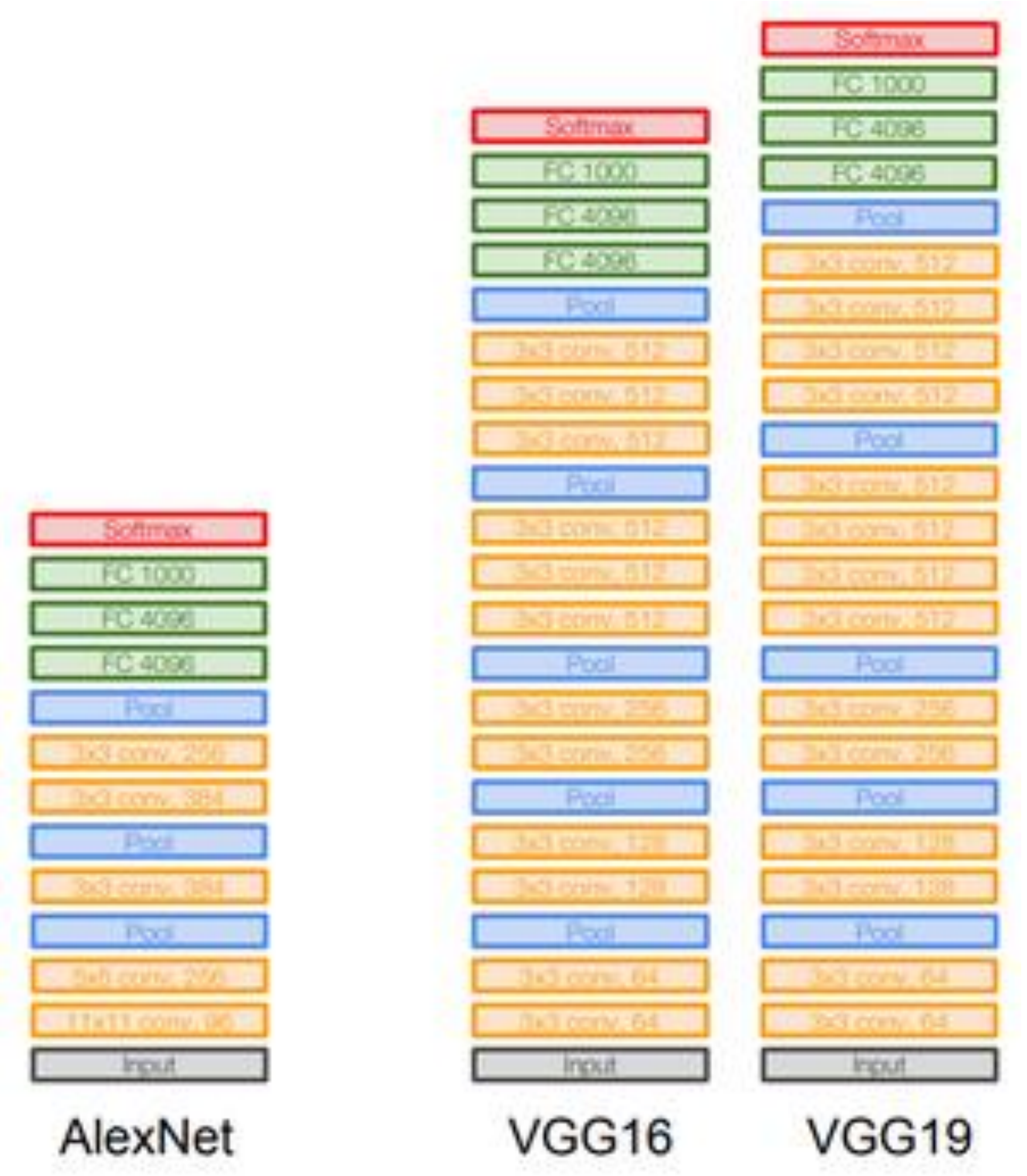
- The VGG network, introduced in 2014 by the Visual Geometry Group at Oxford, offers a deeper yet simpler variant of the convolutional structures discussed above. At the time of its introduction, this model was considered to be very deep.



Very deep convolutional networks for **large**-scale image recognition
K Simonyan, A Zisserman - arXiv preprint arXiv:1409.1556, 2014 - arxiv.org
… In this work we evaluated **very deep convolutional networks** (up to 19 weight layers) for **large**scale image classification. It was demonstrated that the representation depth is beneficial …
Save Cite Cited by 105719 Related articles All 43 versions

VGGNet

- It makes the improvement over AlexNet by *replacing large kernel-sized filters* (11 and 5 in the first and second convolutional layer, respectively) with multiple 3X3 kernel-sized filters one after another.
- It always uses 3 x 3 filters with stride of 1 in convolution layer and uses SAME padding in pooling layers 2 x 2 with stride of 2 – *Simplified architecture*
- *Parameters Learned: 138 Million*



Inception Family

- Prior to its inception, most popular CNNs just stacked convolution layers deeper and deeper, hoping to get better performance.
- The Inception Family is all about going ***wider***.
- The popular versions are as follows:
  - Inception v1 or GoogleNet
  - Inception v2
  - Inception v3
  - Inception v4 and Inception-ResNet

Going deeper with **convolutions**

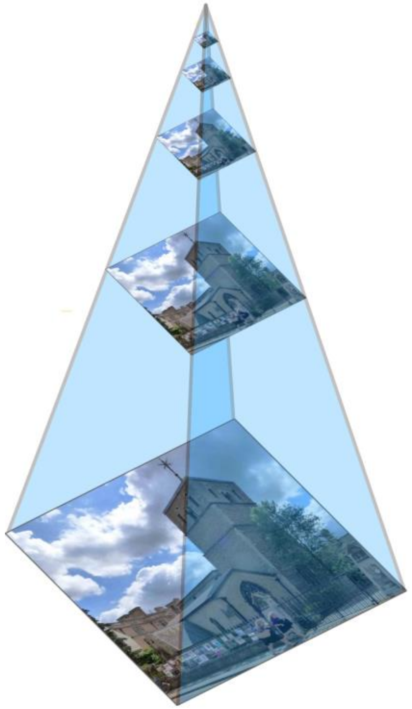
C Szegedy, W Liu, Y Jia, P Sermanet… - Proceedings of the …, 2015 - cv-foundation.org

We propose a **deep** convolutional neural network architecture codenamed Inception that achieves the new state of the art for classification and detection in the ImageNet Large-Scale ...

☆ Save 99 Cite Cited by 51840 Related articles All 57 versions

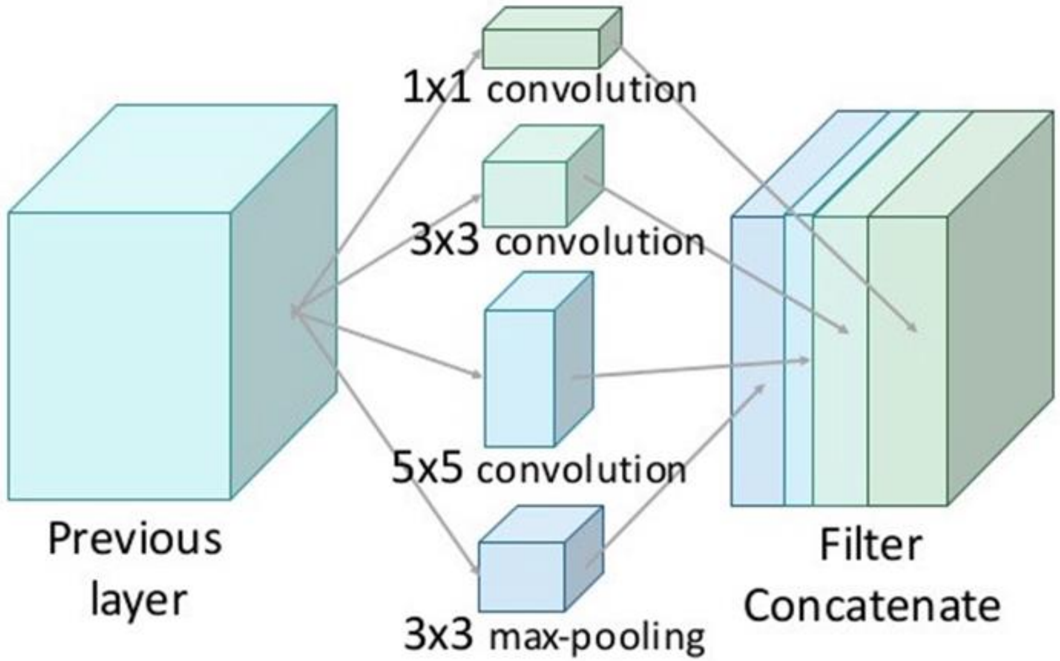
Inception Family

- **Salient parts** in the image can have extremely **large variation** in size. For instance, the area occupied by the dog is different in each image.
- Because of this huge variation in the location of the information, choosing the ***right kernel size*** for the convolution operation becomes tough.
- A ***larger kernel*** is preferred for information that is distributed more ***globally***, and a ***smaller kernel*** is preferred for information that is distributed more ***locally***.



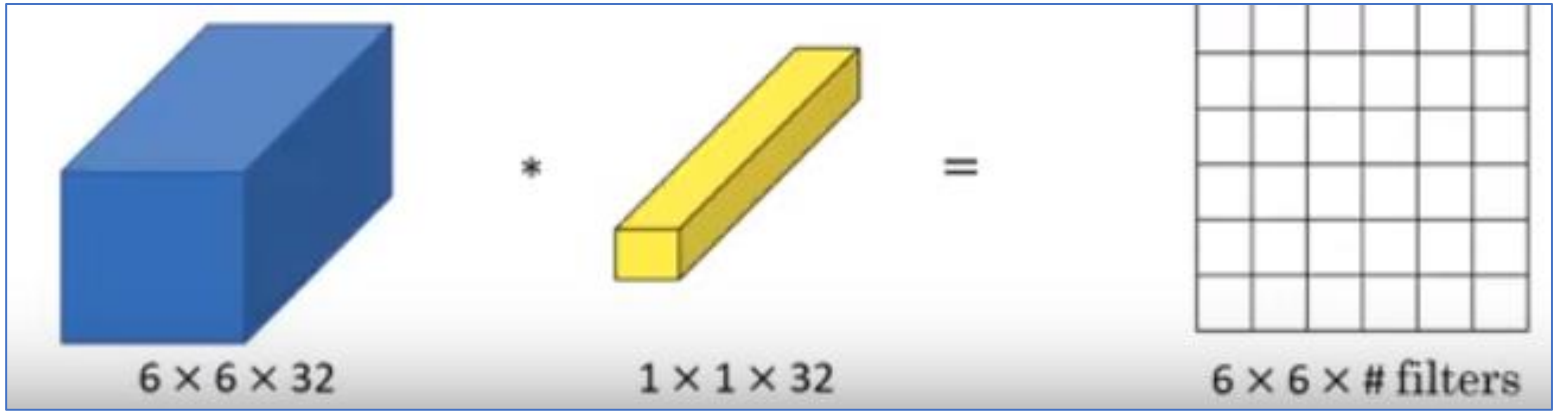
Inception Module

- At any given layer, how do we know what transformation provides the most “useful” information?
- *Why not let the model chose?*
- An Inception module computes *multiple different transformations* over the same input map in parallel, concatenating their results into a single output.
- In other words, for each layer, Inception does a 5x5 convolutional transformation, *and* a 3x3, *and* a max-pool.
- And the next layer of the model gets to decide if (and how) to use each piece of information.



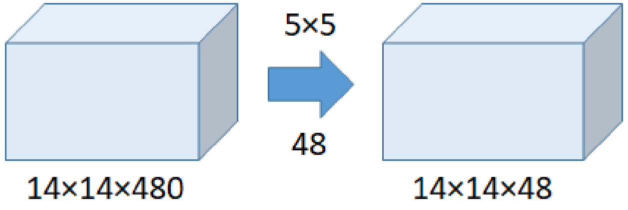
1x1 Convolution

- A 1x1 convolution only looks at one value at a time, but across multiple channels, it can extract spatial information and compress it down to a lower dimension.
- For example, using 20 1x1 filters, an input of size 64x64x100 (with 100 feature maps) can be compressed down to 64x64x20.

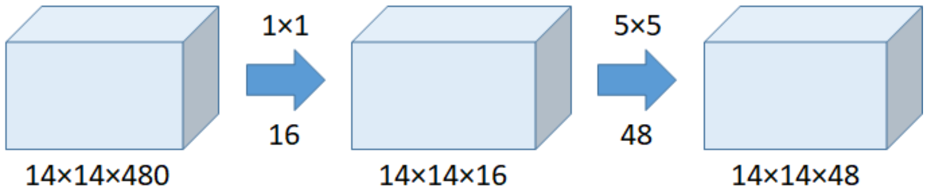


Improved Inception Module

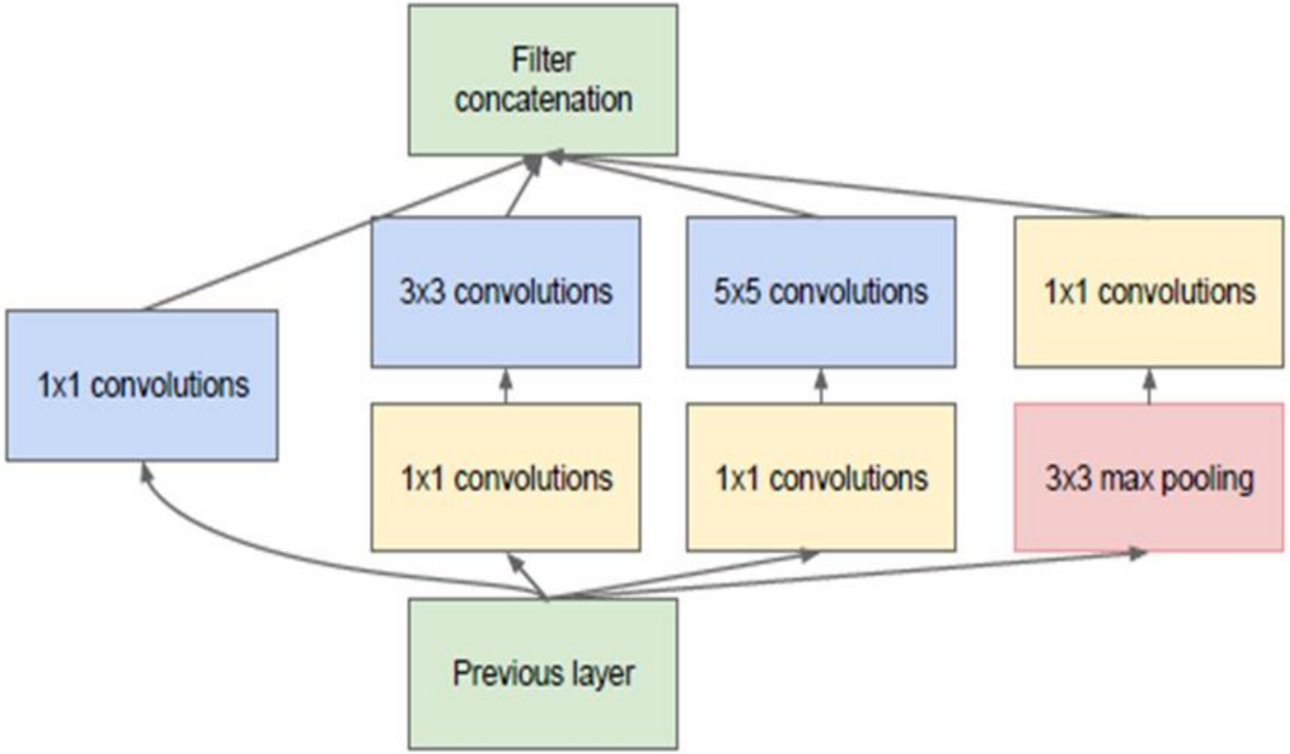
- In GoogLeNet, 1×1 convolution is used as a dimension reduction module to reduce the computation.



Number of operations = (14×14×48)×(5×5×480) = 112.9M

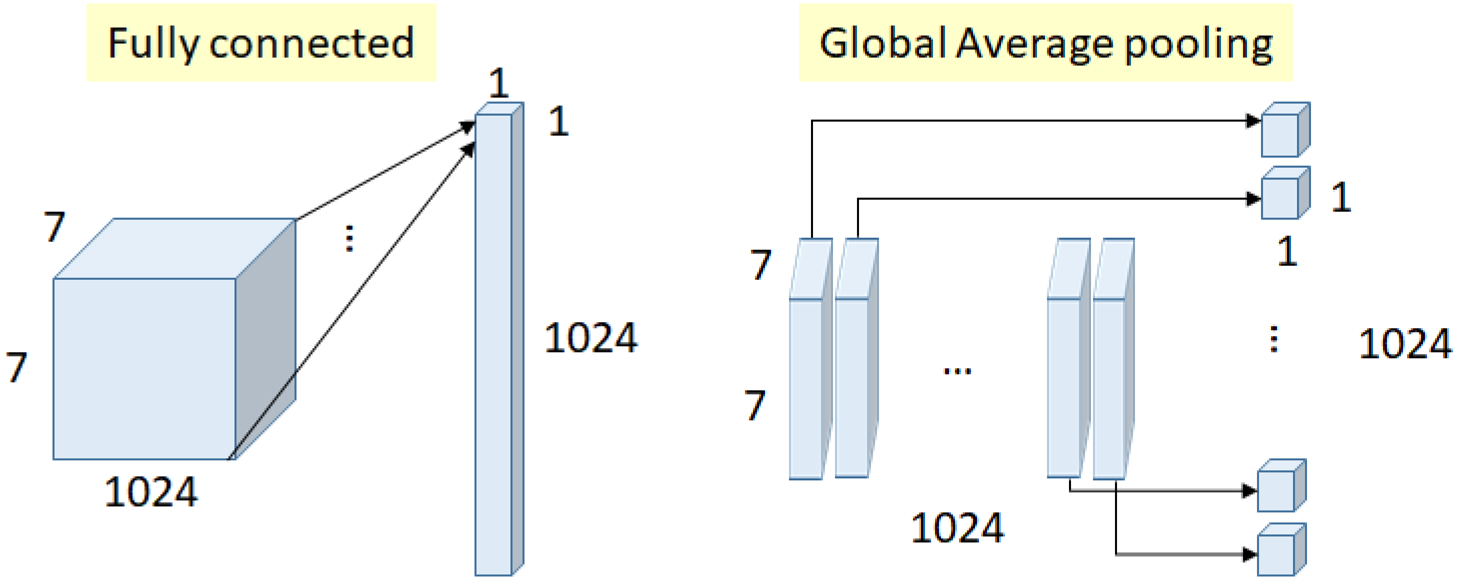


Number of operations for 1×1 = (14×14×16)×(1×1×480) = 1.5M
Number of operations for 5×5 = (14×14×48)×(5×5×16) = 3.8M
Total number of operations = 1.5M + 3.8M = 5.3M



Global Average Pooling

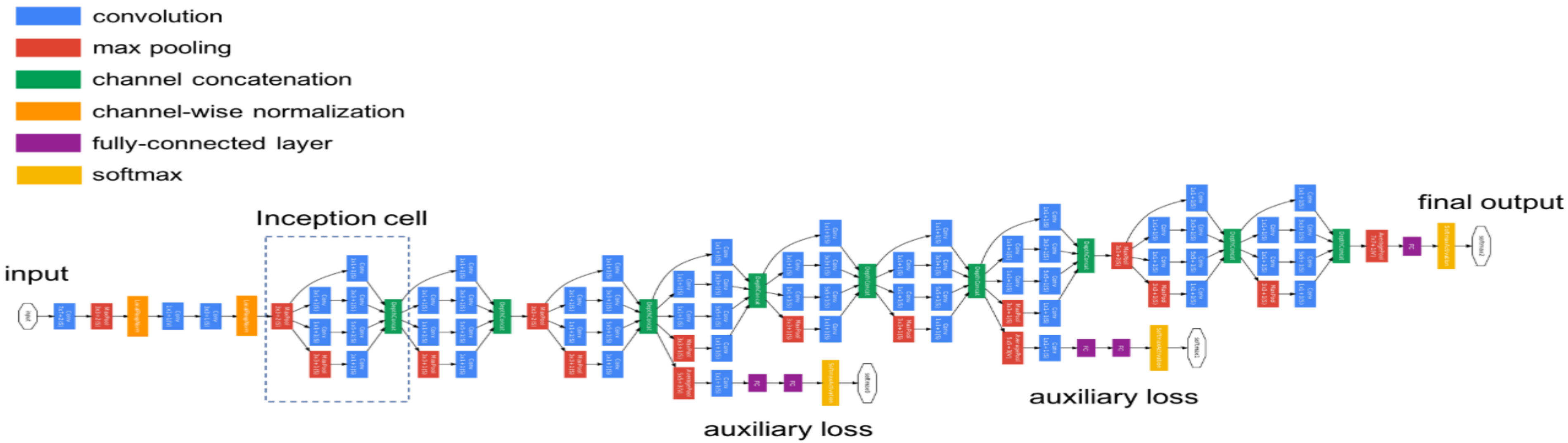
- In GoogLeNet, global average pooling is used at the end of network by averaging each feature map from 7×7 to 1×1, as in the figure below
- GAP layer introduces no parameters,Mean value of each feature map is kept



Lin, Min, Qiang Chen, and Shuicheng Yan. "Network in network." arXiv preprint arXiv:1312.4400 (2013).

Inception-V1 or GoogleNet

- Using the dimension reduced inception module, a neural network architecture was built (2014) which got the *first place in the 2014* ImageNet competition for classification and detection challenges.
- GoogLeNet has 9 such inception modules stacked linearly.
- It is *22 layers* deep (27, including the pooling layers).
- It achieved a top-5 error rate with of 6.67%
- It reduces the number of parameters from 60 million (AlexNet) to 4 million.



Rest of Inception Family

- Inception-v2 introduced **Batch Normalization**

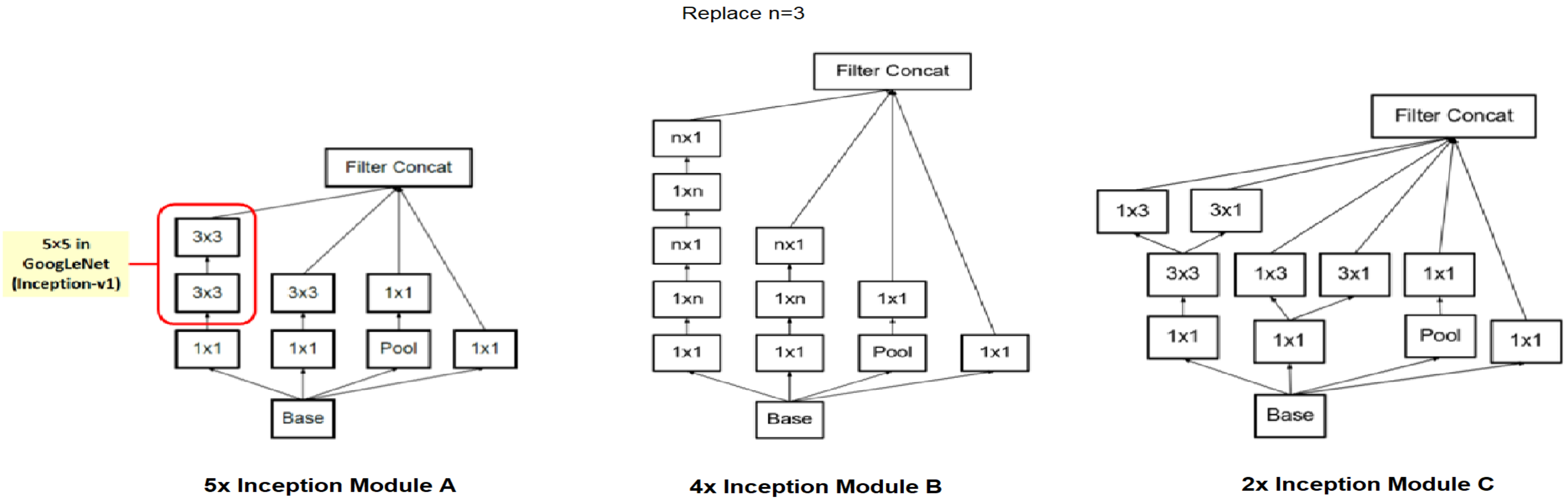
  Paper:

  Batch normalization: Accelerating deep network training by reducing internal covariate shift

- Inception-v3 introduced **Factorized Convolutions**

  Paper:

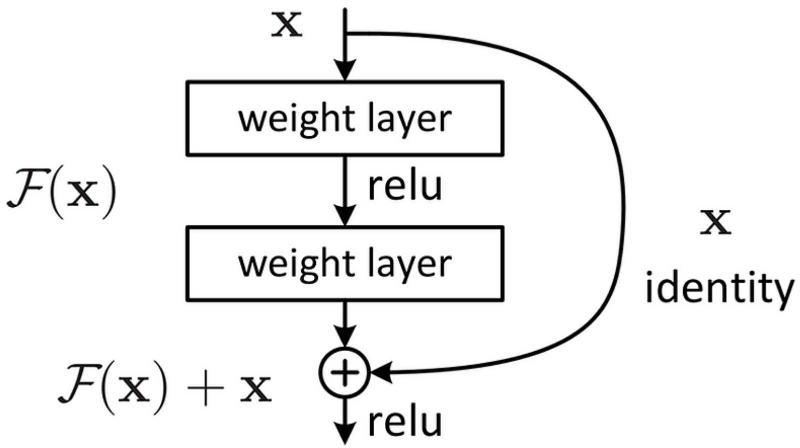
  Rethinking the **inception architecture** for computer vision



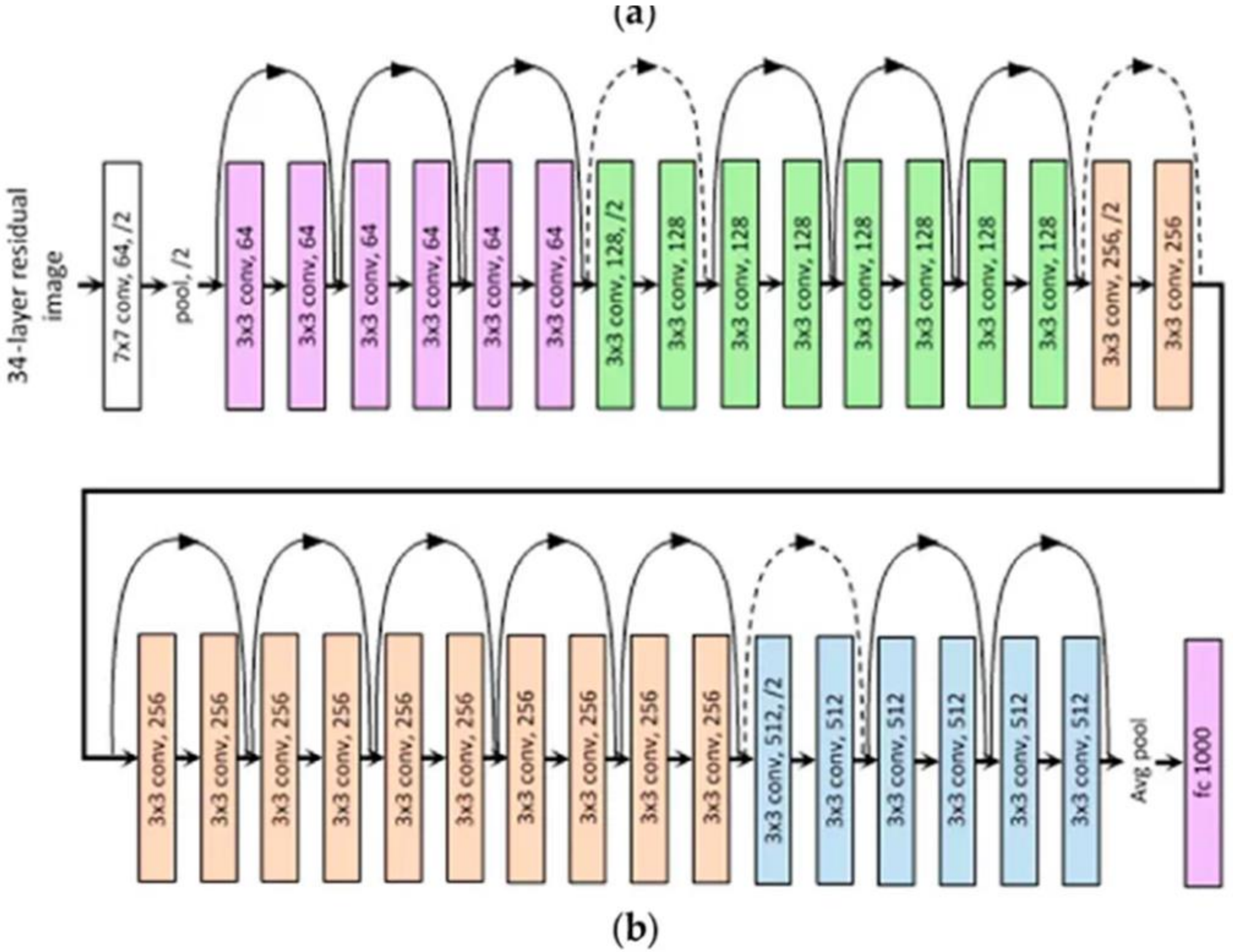
ResNet Architecture

- **Skip Connection:** Residual connections bypass one or more layers and add the input of the skipped layers directly to the output.

Residual Networks



ResNet Architecture



ResNet-34 Layered architecture

ResNet Architecture Advantages

- Solves Vanishing Gradient Problem: By adding inputs directly, residuals help gradients flow back through the network during training, preventing gradients from vanishing in very deep networks.

- Enables Deeper Networks: With residual connections, deeper networks can perform better because they learn “residual” functions.

Other Architectures

- Inception V4
- DenseNet
- EfficientNet

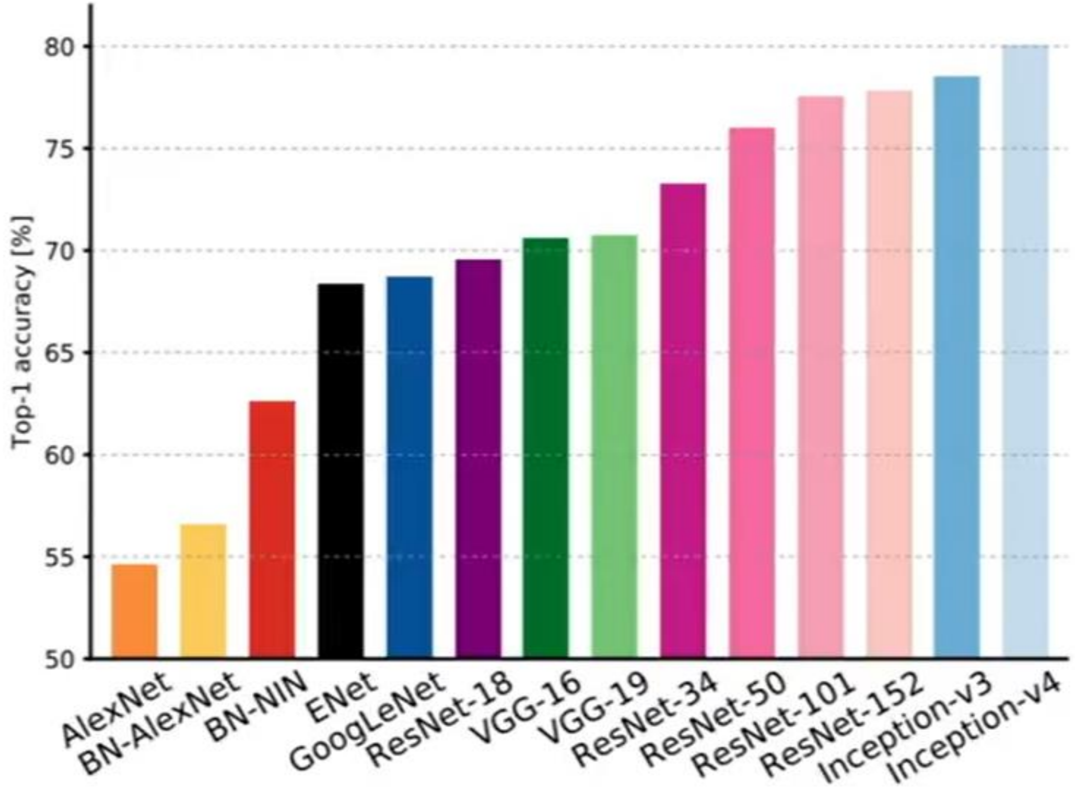
Inception-v4, **inception-resnet** and the impact of residual connections on learning
C Szegedy, S Ioffe, V Vanhoucke, A Alemi - arXiv preprint arXiv ..., 2016 - arxiv.org
... 20 40 60 80 100 120 140 160 Epoch 3 **4** 5 6 7 8 9 Error (top-5) % **inception-v4** **inception-resnet**-v2 Figure 24. Top-5 error evolution during training of pure **Inception-v4** vs a residual **Inception** of similar computational cost ...
Cited by 5813 Related articles All 22 versions

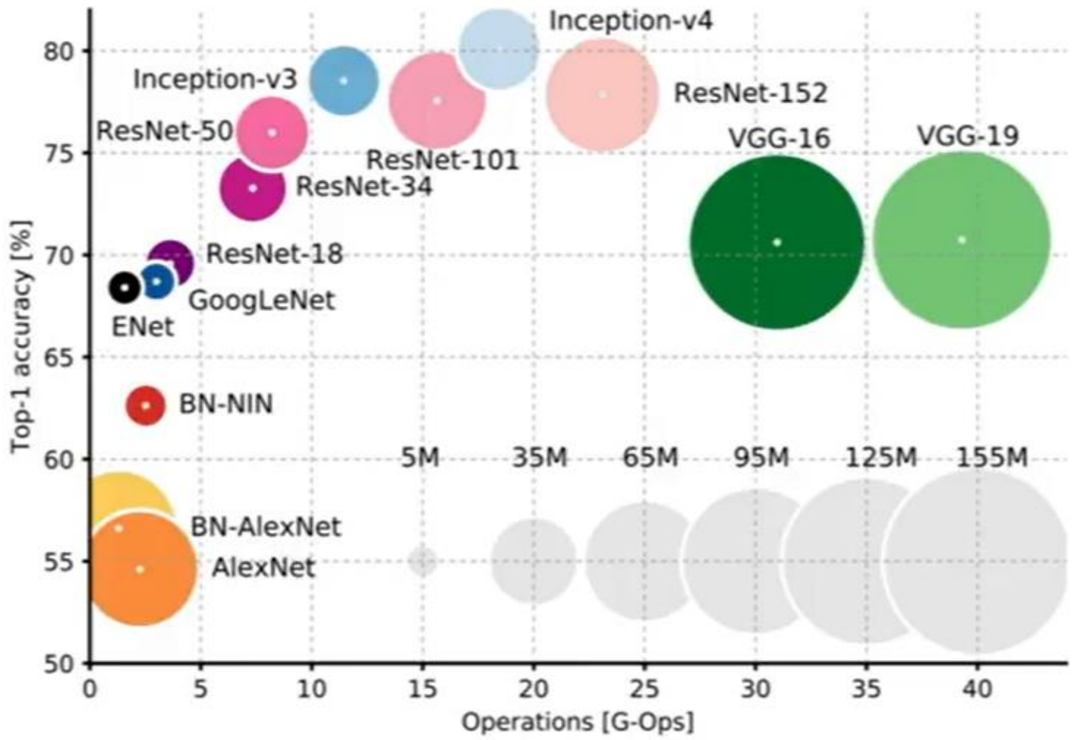
Deep residual learning for image recognition
K He, X Zhang, S Ren, J Sun - Proceedings of the IEEE ..., 2016 - openaccess.thecvf.com
Deeper neural networks are more difficult to train. We present a **residual learning** framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as **learning residual** functions with reference to the layer ...
Cited by 60755 Related articles All 68 versions

Densely connected convolutional networks
G Huang, Z Liu, L Van Der Maaten... - Proceedings of the ..., 2017 - openaccess.thecvf.com
Recent work has shown that convolutional networks can be substantially deeper, more accurate, and efficient to train if they contain shorter connections between layers close to the input and those close to the output. In this paper, we embrace this observation and introduce ...
Cited by 12256 Related articles All 30 versions

https://keras.io/api/applications/

Comparison





**Bigger the circle, greater the memory (parameter) requirements*

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

MNIST Dataset

Explore them on your own!

- Dataset of handwritten digits
- Contains a training set of 60,000 examples and a test set of 10,000 examples
- Size: ~50MB, 70,000 images in 10 classes

Cifar-10 Dataset

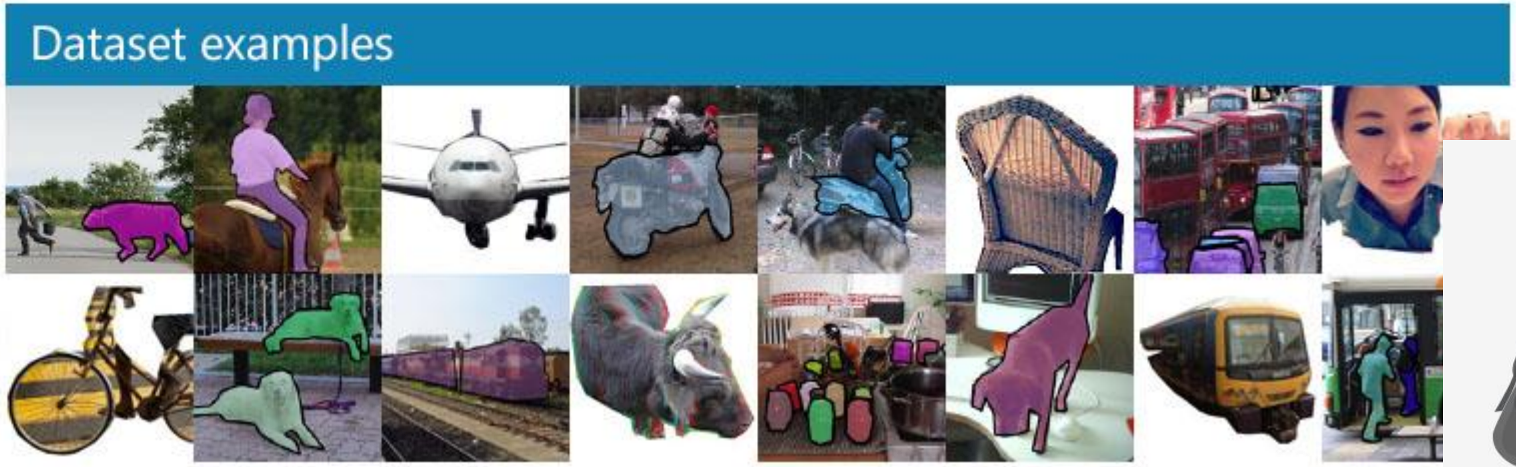
- Size: ~170MB, 60,000 images in 10 classes

ImageNet Dataset

- More than 14 Million Images
- More than 20,000 categories
- Annotated for object categories and (a subset) with object locations
- Different subsets employed in the *Large Scale Visual Recognition Challenge (ILSVRC)*

MS-Coco Dataset

- COCO is a large-scale and rich for object detection, segmentation and captioning
- **Size:** ~25 GB (Compressed)
- **Number of Records:** 330K images, 80 object categories, 5 captions per image, 250,000 people with key points



Training a CNN Model from Scratch

You could have explored that already in the lab!

- Prepare Data (Train/Test Splits)
- Compile Model
- Train Model
- Test Model

Training a CNN Model from Scratch

You could have explored that already in the lab!

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', name='conv_1',
                 input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2), name='maxpool_1'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', name='conv_2'))
model.add(MaxPooling2D((2, 2), name='maxpool_2'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_3'))
model.add(MaxPooling2D((2, 2), name='maxpool_3'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_4'))
model.add(MaxPooling2D((2, 2), name='maxpool_4'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu', name='dense_1'))
model.add(Dense(128, activation='relu', name='dense_2'))
model.add(Dense(1, activation='sigmoid', name='output'))
```

Preparing Data

You could have explored that already in the lab!

```
import numpy as np
import mnist

train_images = mnist.train_images()
train_labels = mnist.train_labels()
test_images = mnist.test_images()
test_labels = mnist.test_labels()

# Normalize the images.
train_images = (train_images / 255) - 0.5
test_images = (test_images / 255) - 0.5

# Reshape the images.
train_images = np.expand_dims(train_images, axis=3)
test_images = np.expand_dims(test_images, axis=3)

print(train_images.shape) # (60000, 28, 28, 1)
print(test_images.shape)  # (10000, 28, 28, 1)
```

Compile Model

You could have explored that already in the lab!

```
model.compile(
  'adam',
  loss='categorical_crossentropy',
  metrics=['accuracy'],
)
```

Train Model

You could have explored that already in the lab!

- The **training data** (images and labels), commonly known as X and Y, respectively.
- The **number of epochs** (iterations over the entire dataset) to train for.
- The **validation data** (or test data), which is used during training to periodically measure the network's performance against data it hasn't seen before.

```
model.fit(
  train_images,
  to_categorical(train_labels),
  epochs=3,
  validation_data=(test_images, to_categorical(test_labels)),
)
```

```
Epoch 1
loss: 0.2433 - acc: 0.9276 - val_loss: 0.1176 - val_acc: 0.9634
Epoch 2
loss: 0.1184 - acc: 0.9648 - val_loss: 0.0936 - val_acc: 0.9721
Epoch 3
loss: 0.0930 - acc: 0.9721 - val_loss: 0.0778 - val_acc: 0.9744
```

Test Model

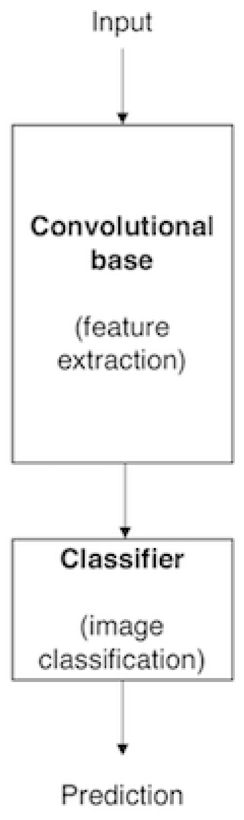
You could have explored that already in the lab!

```
model.load_weights('cnn.h5')
```

```
# Predict on the first 5 test images.
predictions = model.predict(test_images[:5])
```

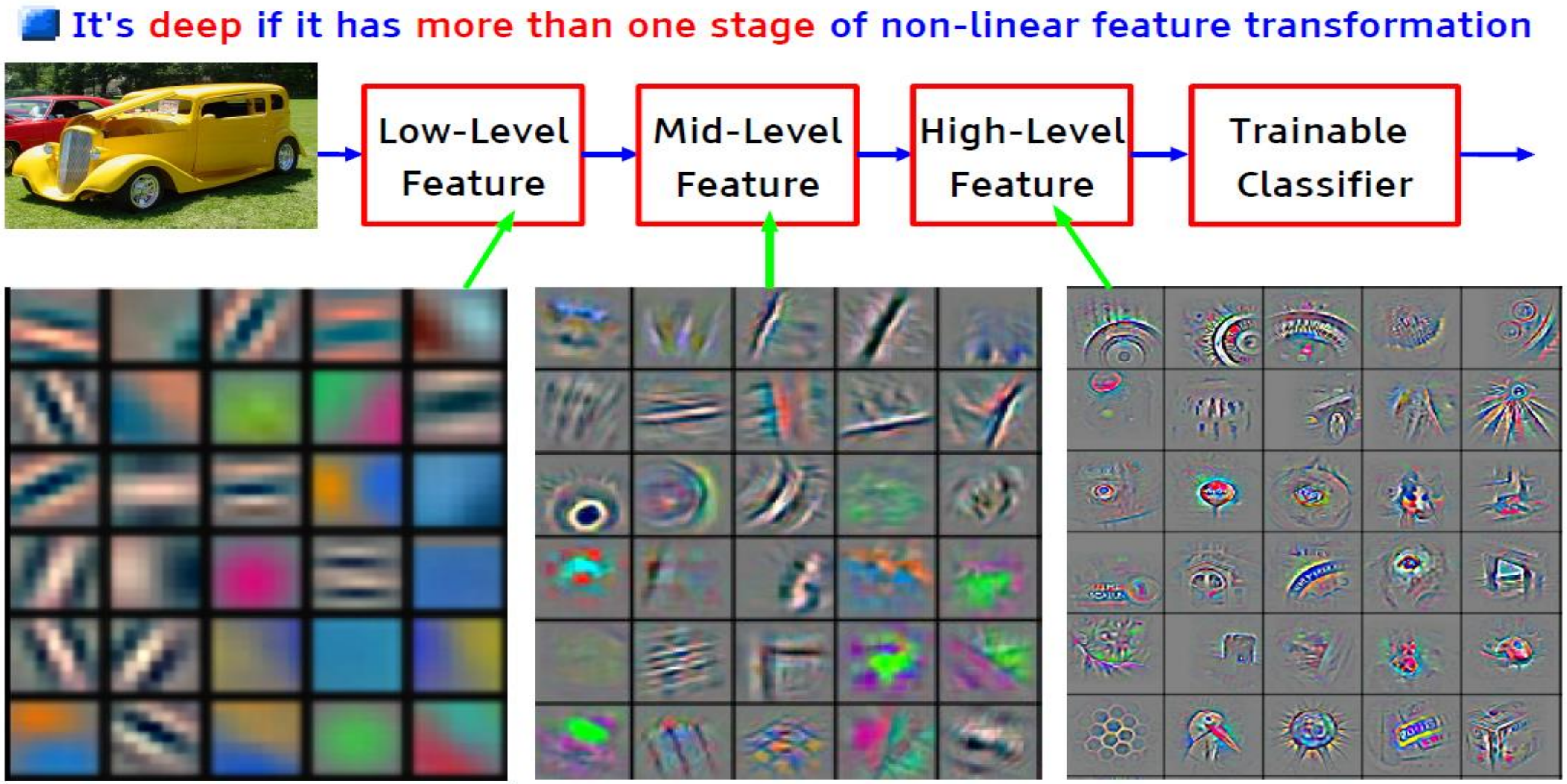
Transfer Learning

- A Typical CNN has two parts:
  - **Convolutional base**: which is composed by a stack of convolutional and pooling layers. The main goal of the convolutional base is to generate features from the image.
  - **Classifier**: which is usually composed by fully connected layers. The main goal of the classifier is to classify the image based on the detected features. A fully connected layer is a layer whose neurons have full connections to all activation in the previous layer.



Using Pre-Trained Nets

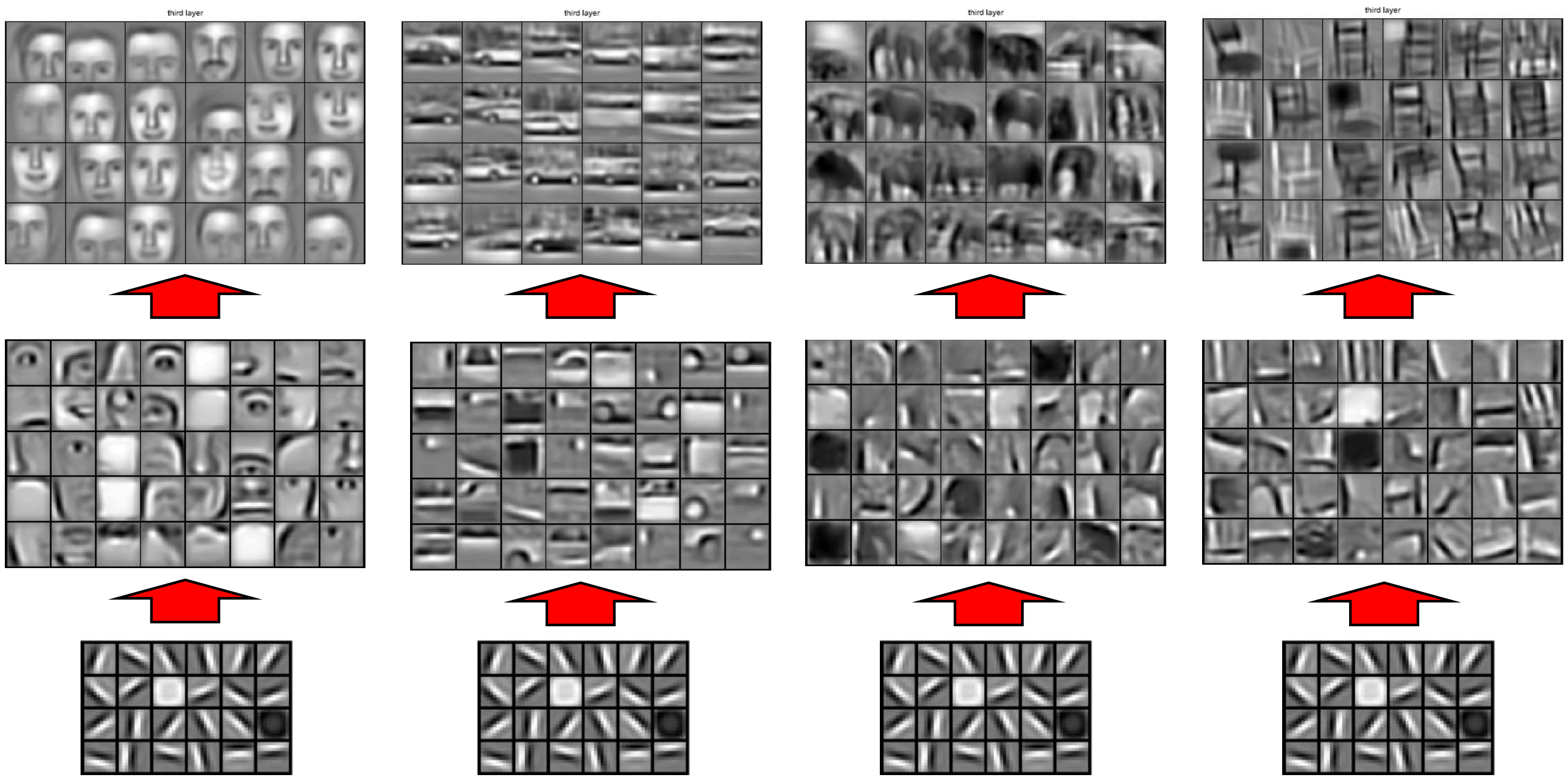
- Deep learning models can automatically learn ***hierarchical feature representations***.



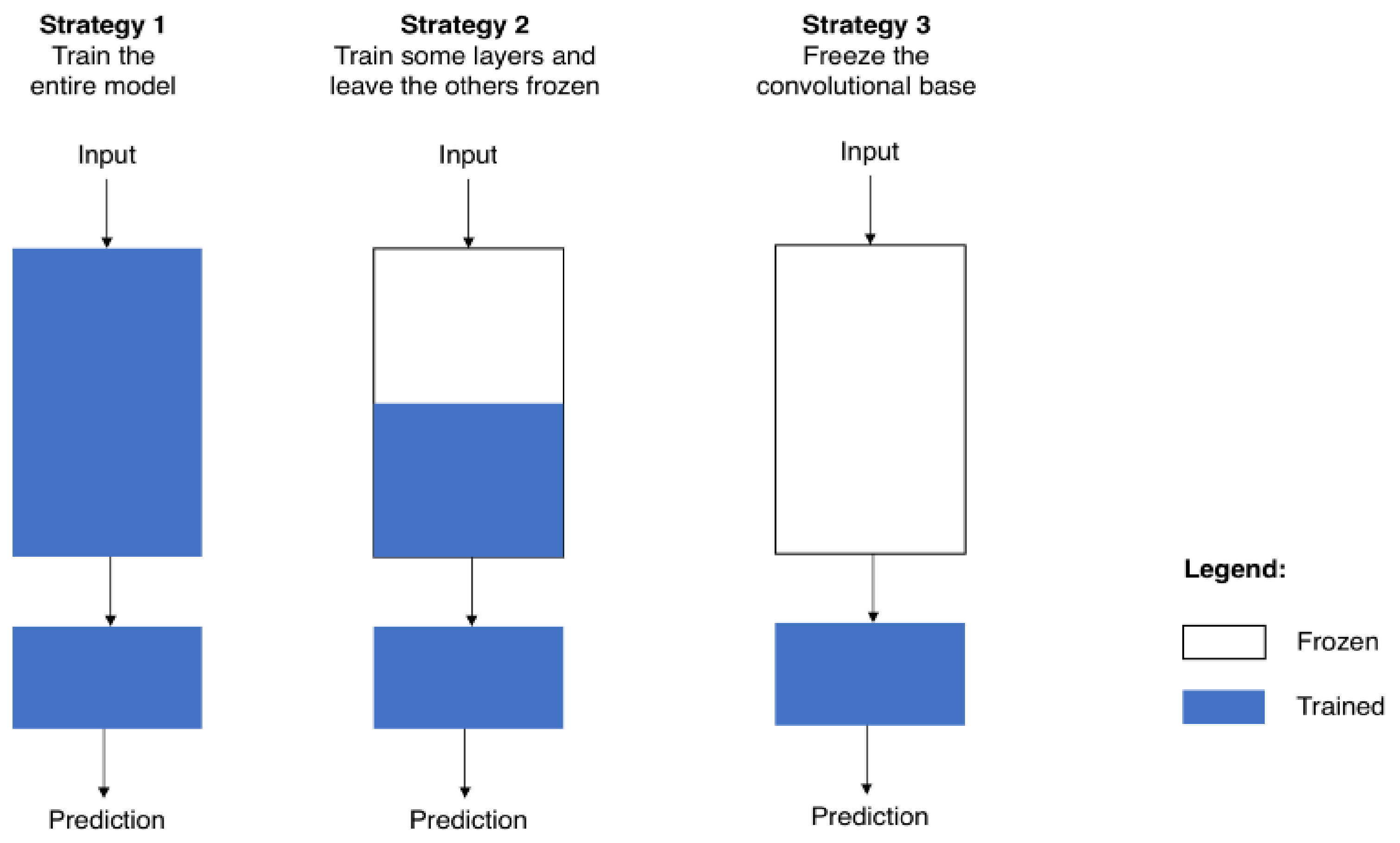
Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Using Pre-Trained Nets

- Features computed by the first layer are general and can be reused in different problem domains, while features computed by the last layer are specific and depend on the chosen dataset and task
- ***Transfer Learning: Taking a pre-trained model and adapting it to a given problem.***



Using Pre-Trained ConvNets



Credits: Transfer Learning using Pre-trained models, Pedro Marcelino

Using Pre-Trained Nets

- **Strategy I: Train the entire model**

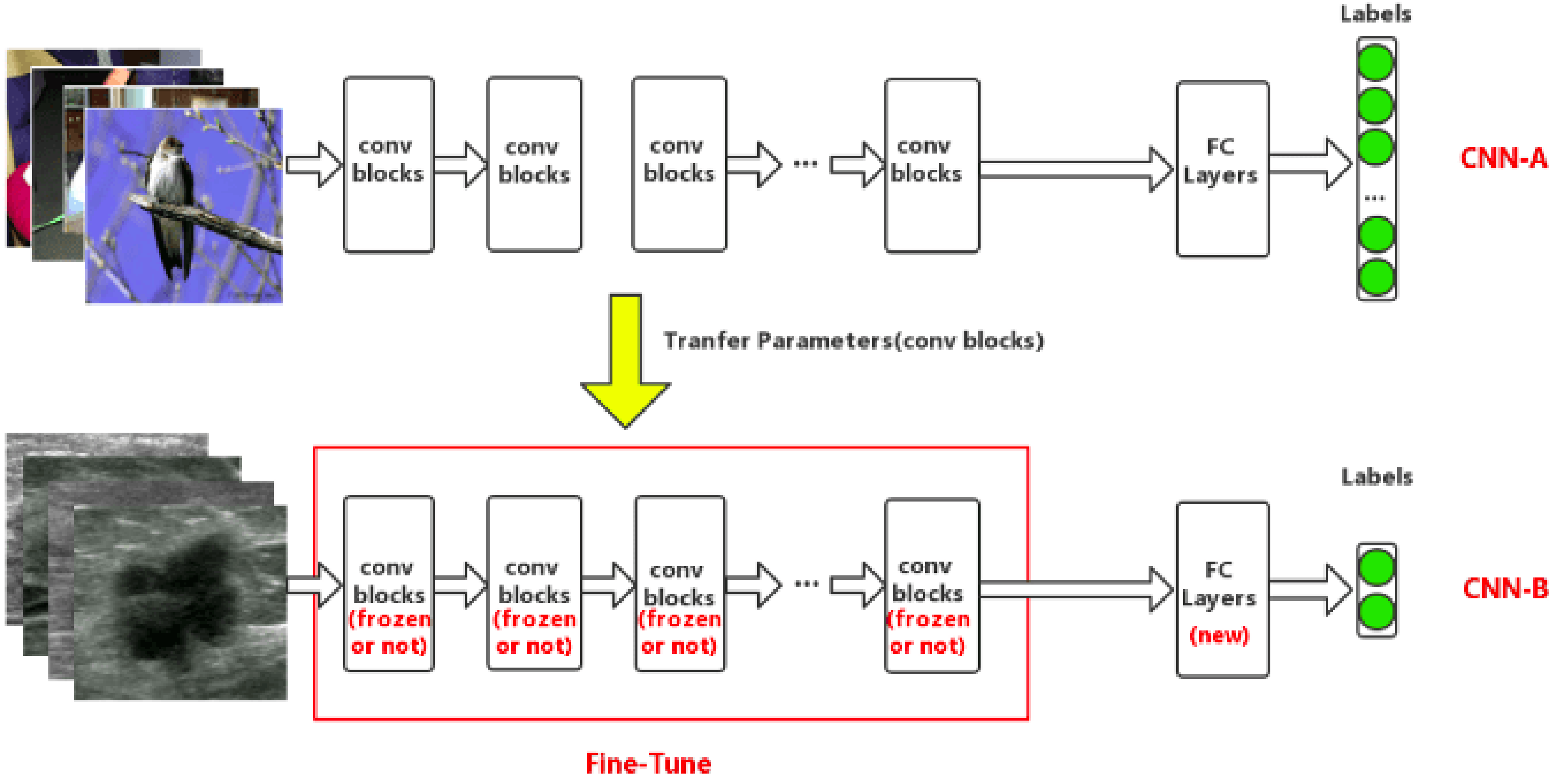
    - In this case, you use (only) the architecture of the pre-trained model and train it according to your dataset.

    - You're learning the model from scratch, so you'll need a large dataset (and a lot of computational power).

Using Pre-Trained Nets

- **Strategy II: Fine-Tune a pre-trained model**
  - Change the fully connected layer of the network to match the data under study
  - Continue back propagation and update parameters of all or a subset of layers (Initial layers can be frozen)

Fine-Tuning

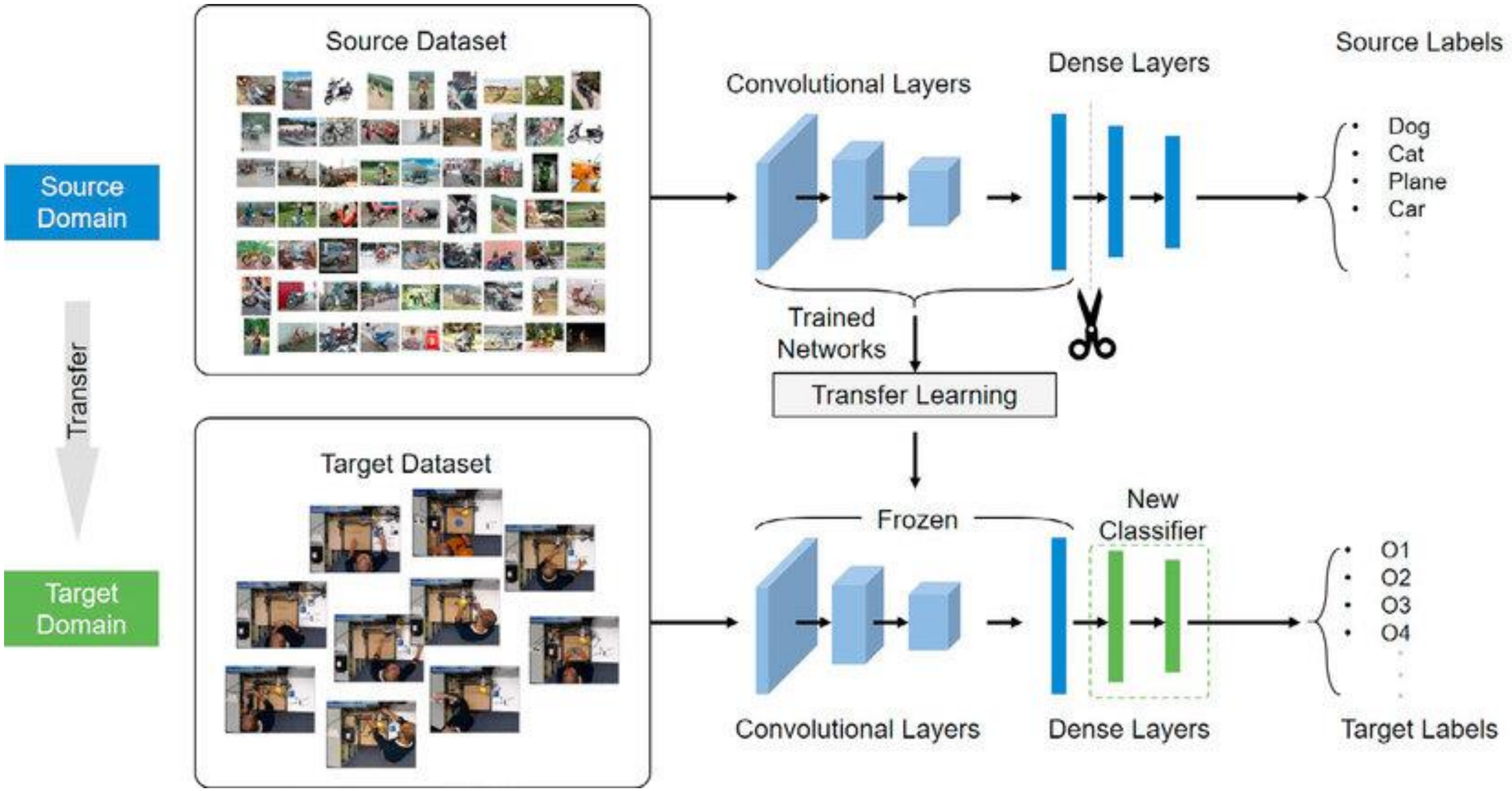


Labels
conv blocks
conv blocks
conv blocks
...
conv blocks
FC Layers
CNN-A
Tranfer Parameters(conv blocks)
Labels
conv blocks (frozen or not)
conv blocks (frozen or not)
conv blocks (frozen or not)
...
conv blocks (frozen or not)
FC Layers (new)
CNN-B
Fine-Tune

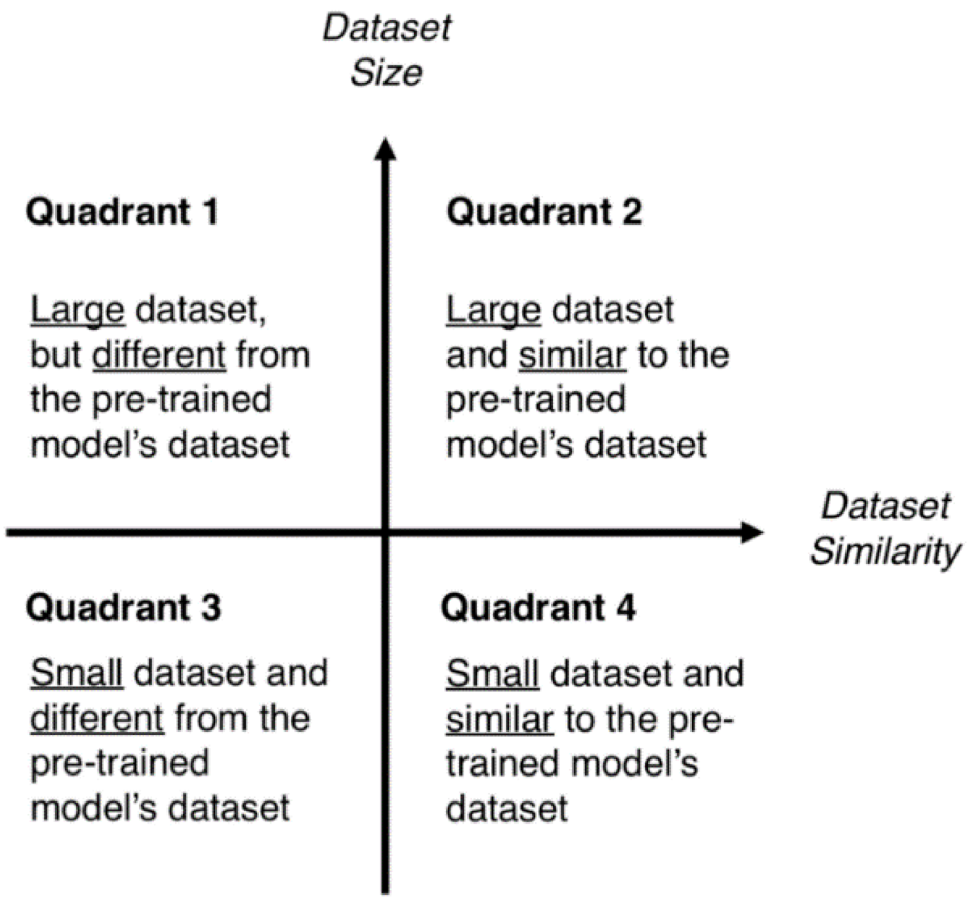
Using Pre-Trained Nets

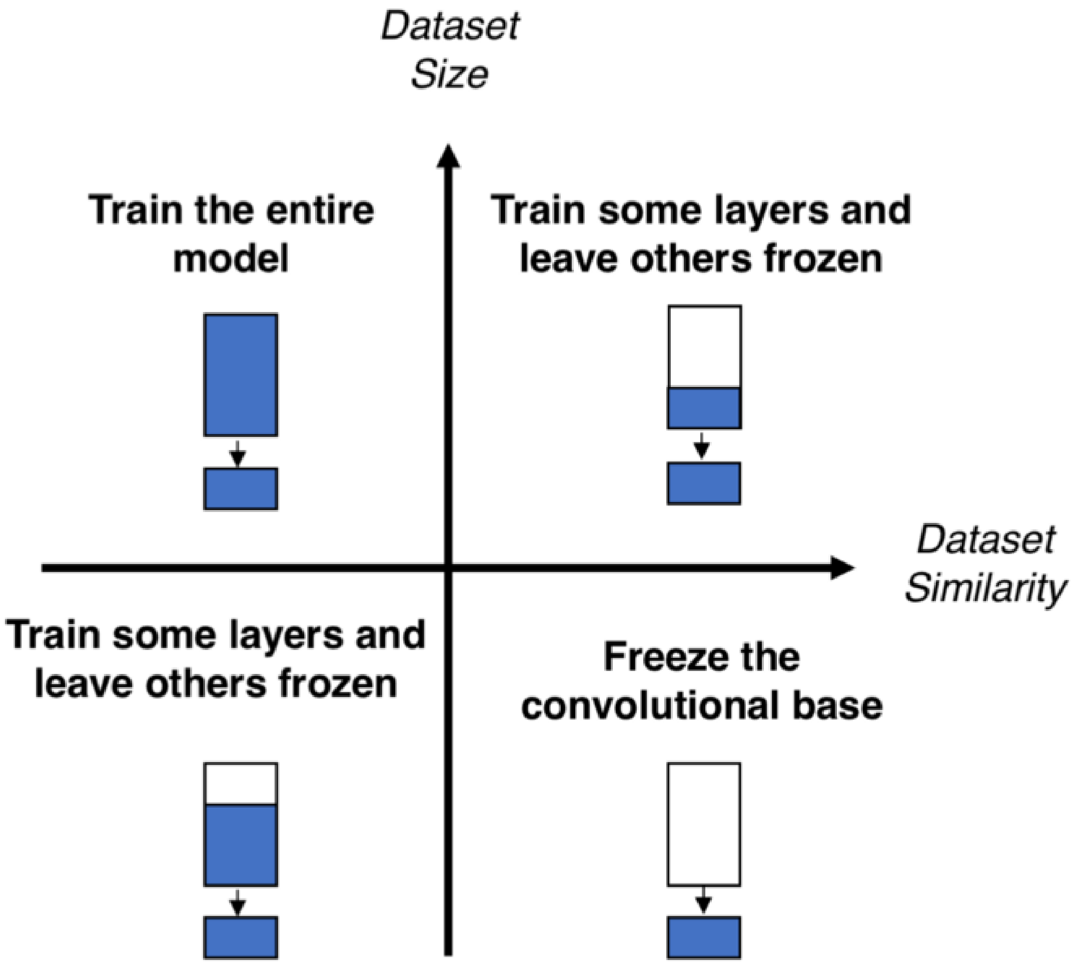
- **Strategy III: Use CNN as Feature Extractor**
  - Freeze the convolutional base
  - Pass the data through network and use the output of convolutional base as features
  - Feed features to another classifier
  - **Example:**
    - **For AlexNet:** 4096-D vector for every image that contains the activations of the hidden layer immediately before the classifier.
    - Train Classifier on these features (e.g. SVM)

Pretrained ConvNets as Feature Extractors



When to use What?





Credits: Transfer Learning using Pre-trained models, Pedro Marcelino

Using Pretrained – Feature Extraction

You will explore that in the lab!

- Load the model:

```
#Include_top=False, Does not load the last two fully connected layers which act as the classifier.
#We are just loading the convolutional layers.
#It should be noted that the last layer has a shape of 7 x 7 x 512.
vgg_conv = VGG16(weights='imagenet',include_top=False,input_shape=(224,224,3))
```

- Load Data and Labels

```
X_train,X_test,Y_train,Y_test = train_test_split(data,labels, test_size=0.20, random_state=42)

X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
```

Using Pretrained – Feature Extraction

You will explore that in the lab!

- Convert labels to one hot encoding

```
# convert class vectors to binary class matrices
Y_train = keras.utils.to_categorical(Y_train, num_classes)
Y_test = keras.utils.to_categorical(Y_test, num_classes)
```

- Create tensors to store features

```
nTrain = X_train.shape[0]
nVal = X_test.shape[0]
train_features = np.zeros(shape=(nTrain,7,7,512))
val_features = np.zeros(shape=(nVal,7,7,512))
```

Using Pretrained – Feature Extraction

- Pass images through network using predict function to get features

```
train_features = vgg_conv.predict(X_train)
train_features = np.reshape(train_features, (nTrain, 7 * 7 * 512))

val_features = vgg_conv.predict(X_test)
val_features = np.reshape(val_features, (nVal, 7 * 7 * 512))
```

- Employ any classifier: Feed it with training features and respective labels

Using Pretrained – Fine Tuning

You will explore that in the lab!

- Load the model

```
vgg_conv = VGG16(weights='imagenet',include_top=False,input_shape=(224,224,3))
```

- Freeze the initial layers

```
# Freeze the layers except the last 4 layers
for layer in vgg_conv.layers[:-4]:
    layer.trainable = False
```

Using Pretrained – Fine Tuning

You will explore that in the lab!

- Create new model: Add classification layers on top of convolutional base

```
from keras import models
from keras import layers
from keras import optimizers

# Create the model
model = models.Sequential()

# Add the vgg convolutional base model
model.add(vgg_conv)

# Add new layers
model.add(layers.Flatten())
model.add(layers.Dense(1024, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(3, activation='softmax'))
```

Acknowledgements

Some of the content of these slides is taken from:

- www.coursera.com
- Slides by CS231n Winter 2016 – Andrej Karpathy
- Convolutional Neural Networks (CNNs): An Illustrated Explanation, Abhineet Saxena
- An Intuitive Explanation of Convolutional Neural Networks
- A Beginner's Guide To Understanding Convolutional Neural Networks, Adit Deshpande
- Deep Learning Andrew Ng