CS195: Computer Vision

Popular CNNs (AlexNet, VGG-16, ResNet)
Dissecting CNNs

Monday, September 23rd, 2024

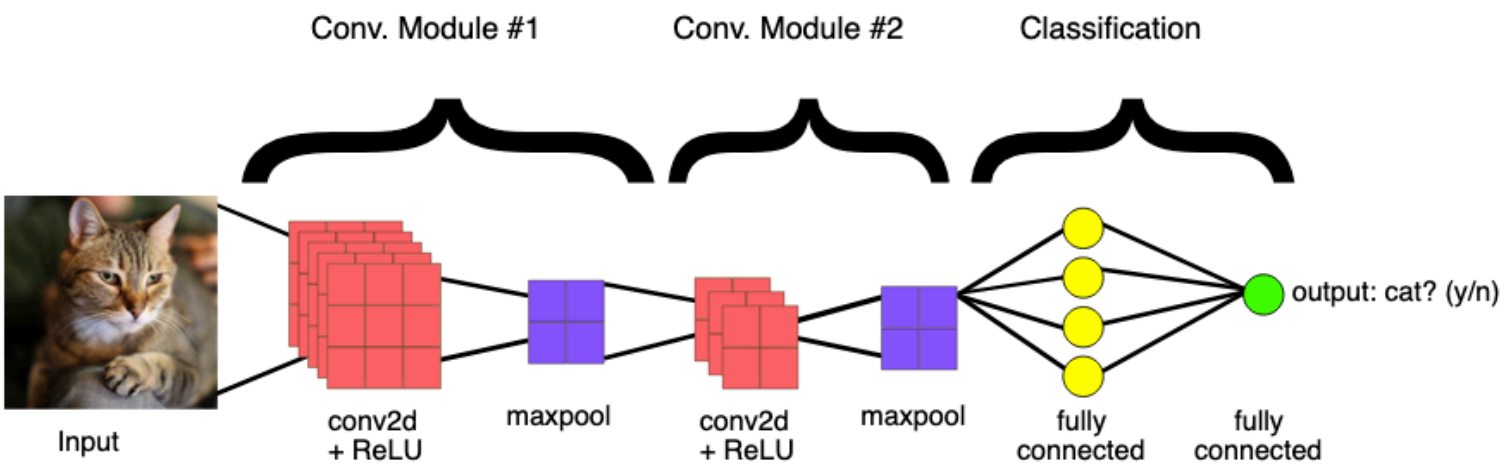
Drake
UNIVERSITY

Today's Agenda

- Popular CNNs
  - LeNet
  - AlexNet
  - VGG
  - ResNet

CNN: A Composition of Convolutional Layers

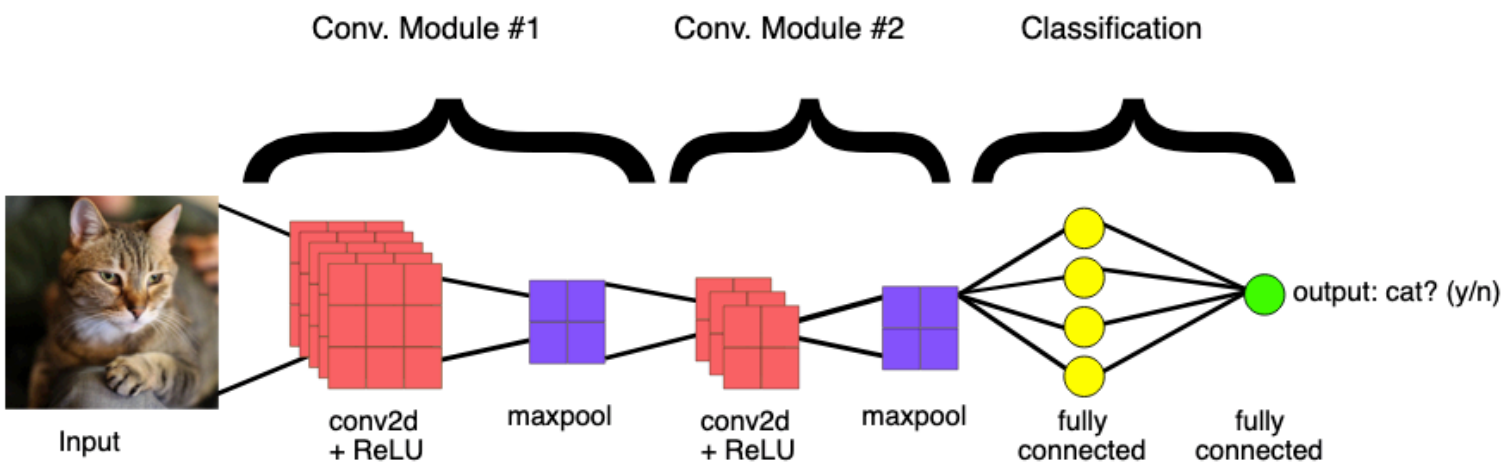
- We've talked about **image data**, **convolutions**, **nonlinearity**, **max pooling**, and how they are related to some computer vision tasks. Let's connect the dots
  - input is an image (in this case a color image, so 3 channels—red, green, and blue)
  - there are several filters, not just one.
  - `Conv2D` layers with `ReLU` are often followed by `maxpool`
  - towards the end of the model, we switch to fully connected (`Dense`) layer
  - We have as many output nodes as we have classes to predict



Reference

CNN: A Composition of Convolutional Layers

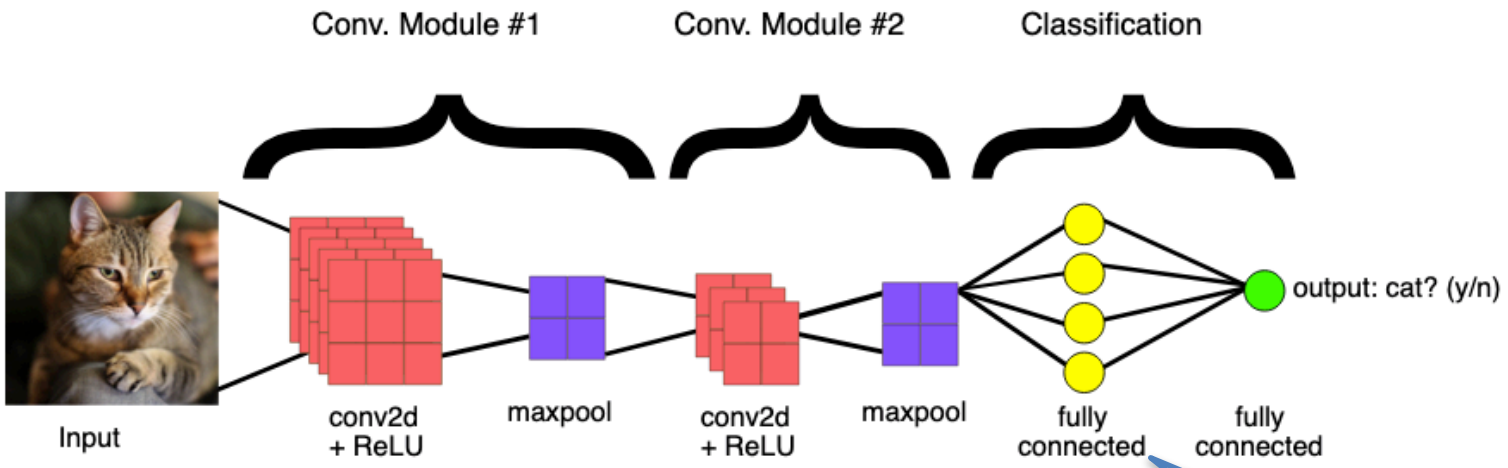
- **Big idea**: different kernels/filters can be used to extract specific information from the original image
- **Bigger idea**: Instead of using manually made kernels for feature extraction, through **deep CNNs we can learn these kernel values** (just like the weights of a traditional NN). These kernels can extract latent features
  - In MLP the way we learn is by changing the ***weights***
  - In CNNs, the way we learn is by changing the values in the **filters/kernels**



Reference

Convolutional Neural Network (CNN)

- Stack multiple stages of feature extractors
- Higher stages compute more global, more invariant features
- Linear layer (just like an MLP) at the end for the final classification



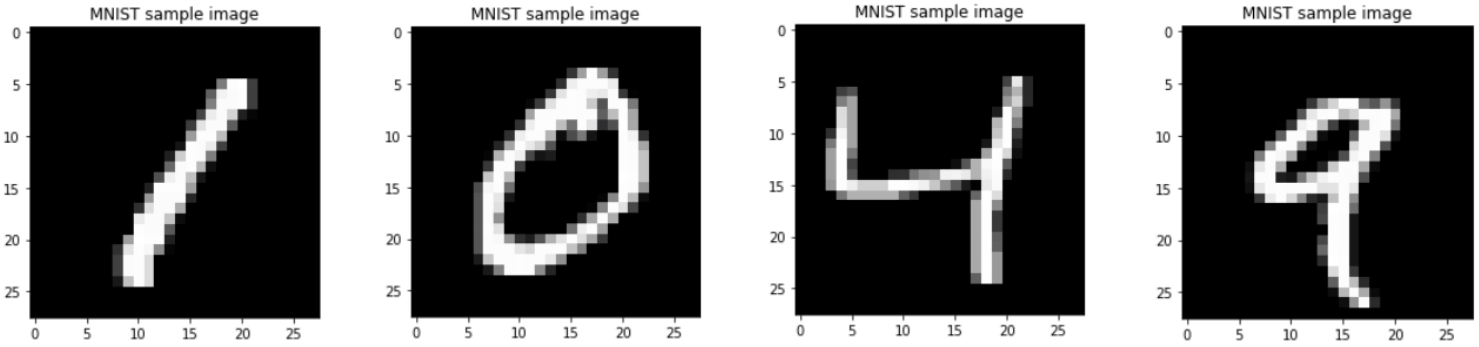
Many names but they refer to the same thing

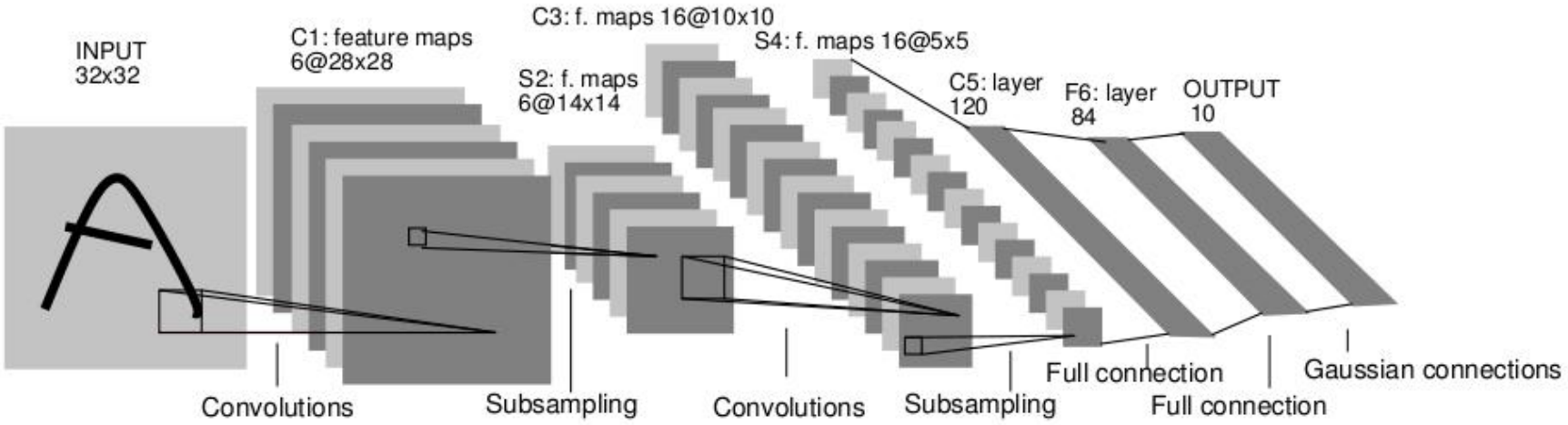
Today's Agenda

- Popular CNNs
  - LeNet
  - AlexNet
  - VGG
  - ResNet

Popular CNN: LeNet

- LeNet is a simple CNN architecture suitable for well-structured image
  - e.g., 28x28 pixels image of digits from 0 to 9 in MNIST or our Fashion-MNIST dataset



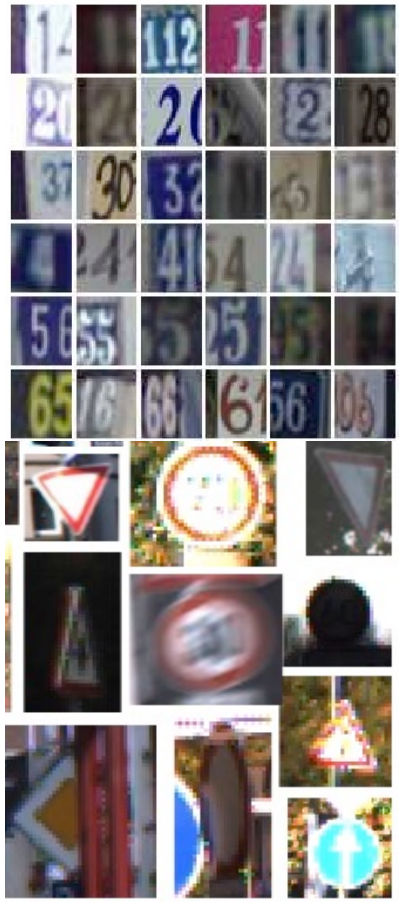


CNNs Success on Simpler Datasets

- Handwritten text/digits
  - MNIST (0.17% error [Ciresan et al. 2011]
  - Arabic & Chinese [Ciresan et al. 2012]

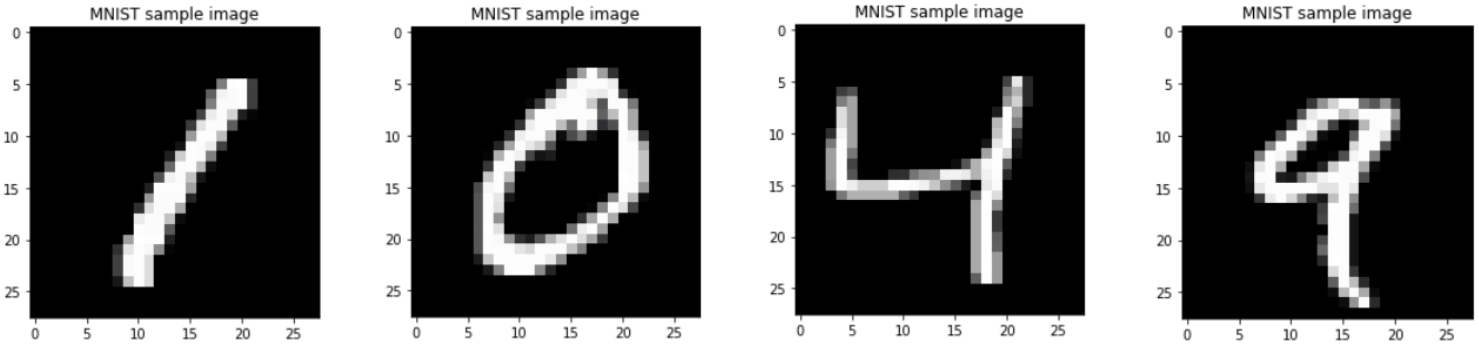
- Simpler recognition benchmarks
  - CIFAR-10 (9.3% error [Wan et al. 2013])
  - Traffic sign recognition

- Until 2011, it was less good at more complex datasets
  - Caltech-101/256 (few training examples)



LeNet insufficient for real-world images

- LeNet is a simple CNN architecture suitable for well-structured image
  - e.g., 28x28 pixels image of digits from 0 to 9 in MNIST or our Fashion-MNIST dataset



- Real-world images are much more complicated; pose challenges in classification
  - e.g., high resolution images 600x480 pixels image and contents have a lot more diversity

ImageNet Challenge

- **ImageNet dataset**
  - 14 million labeled images
  - 20k classes

- **Data source**
  - Images gathered from Internet

- **Image annotators**
  - Human labels via Amazon Turk

Project led by Fei-Fei Li at Stanford
CVPR 2009

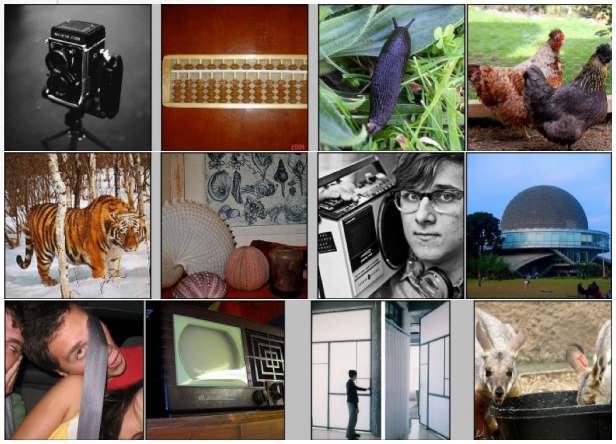
- **What is ImageNet challenge**
  - Train your network with a subset of 1.2 million training images from ImageNet
  - Test your network using another testing subset where you need to classify each image to one of the 1000 classes

ImageNet Challenge

```
imagenet_1000_classes_label_map = \

{0: 'tench, Tinca tinca',
 1: 'goldfish, Carassius auratus',
 2: 'great white shark, white shark, man-eater, man-eating shark, Carcharodon carcharias',
 3: 'tiger shark, Galeocerdo cuvieri',
 4: 'hammerhead, hammerhead shark',
 5: 'electric ray, crampfish, numbfish, torpedo',
 6: 'stingray',
 7: 'cock',
 8: 'hen',
 9: 'ostrich, Struthio camelus',
 10: 'brambling, Fringilla montifringilla',
 11: 'goldfinch, Carduelis carduelis',
 12: 'house finch, linnet, Carpodacus mexicanus',
 13: 'junco, snowbird',
 14: 'indigo bunting, indigo finch, indigo bird, Passerina cyanea',
 15: 'robin, American robin, Turdus migratorius',
 16: 'bulbul',
 17: 'jay',
 18: 'magpie',
 19: 'chickadee',
 20: 'water ouzel, dipper',
```

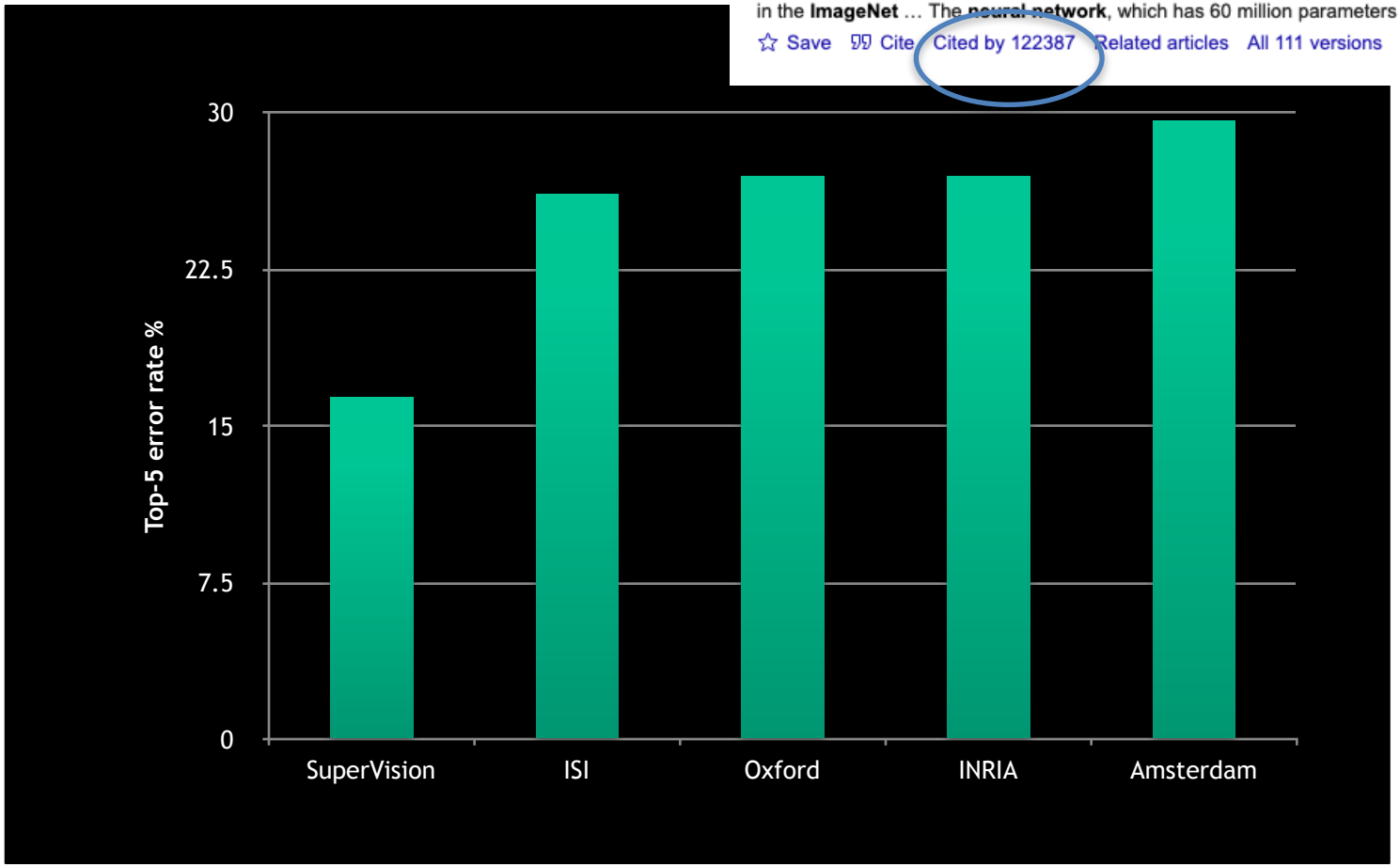
IMAGENET



Project led by Fei-Fei Li at Stanford
CVPR 2009

ImageNet Challenge 2012

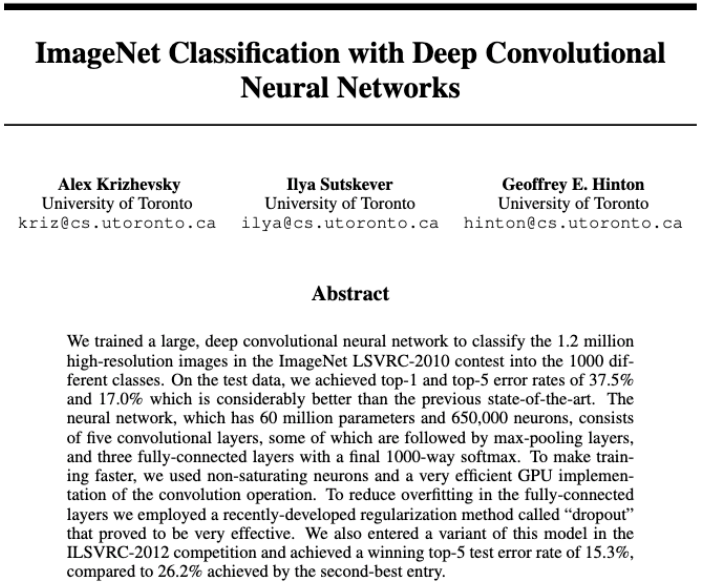
Imagenet classification with deep convolutional neural networks
A Krizhevsky, I Sutskever... - Advances in **neural** ..., 2012 - proceedings.neurips.cc
... a large, **deep convolutional neural network** to **classify** the 1.2 million high-resolution images in the **ImageNet** ... The **neural network**, which has 60 million parameters and 650,000 neurons, ...
☆ Save 99 Cite Cited by 122387 Related articles All 111 versions »



- AlexNet (Krizhevsky et al.) -- **16.4% error** (top-5)
- Next best (non-convnet) – **26.2% error**

Popular CNN: AlexNet

- AlexNet's important features. Similar framework to LeNet with important distinction such as:
  - Bigger model compared to LeNet
    - 5 convolution layers + 3 linear layers
    - 60,000,000 params
  - Trained on more data
    - 10^6 images (ImageNet) vs. 10^3 images (digit imag
  - GPU implementation (50x speedup over CPU)
    - trained on two GPUs for a week
  - Better regularization for training
    - introduced a new technique called **Dropout**
    - **Dropout:** randomly turning off neurons from the network



ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky University of Toronto kriz@cs.utoronto.ca
Ilya Sutskever University of Toronto ilya@cs.utoronto.ca
Geoffrey E. Hinton University of Toronto hinton@cs.utoronto.ca

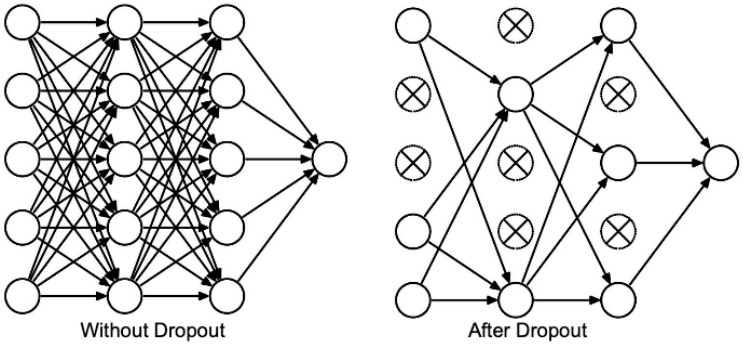
Abstract

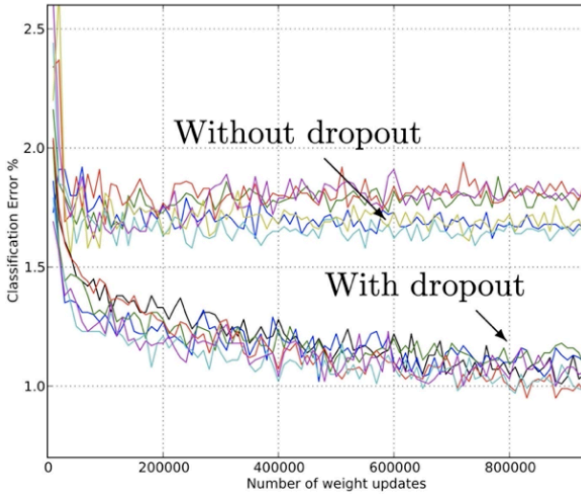
We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

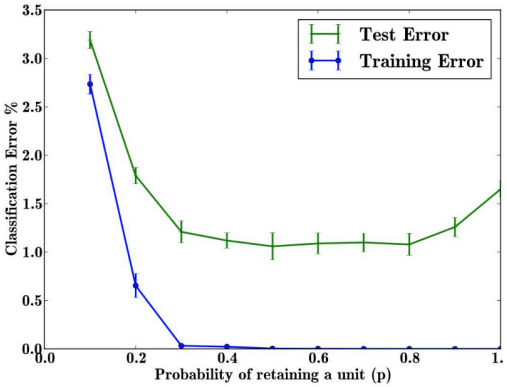
A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Dropout

- AlexNet's important features:
  - Better regularization during training a neural network:
    - introduced a new technique called **Dropout**
    - **Dropout:** randomly set output of a hidden unit to **zero** with a probability **p**
    - **Dropped out unit** has no contribution in forward of backward passes
  - Choosing dropout rate:
    - p: probability of retaining a unit (range 0.5-0.8)
    - n: number of hidden units
    - smaller p leads to under-fitting
    - large p may lead to overfitting

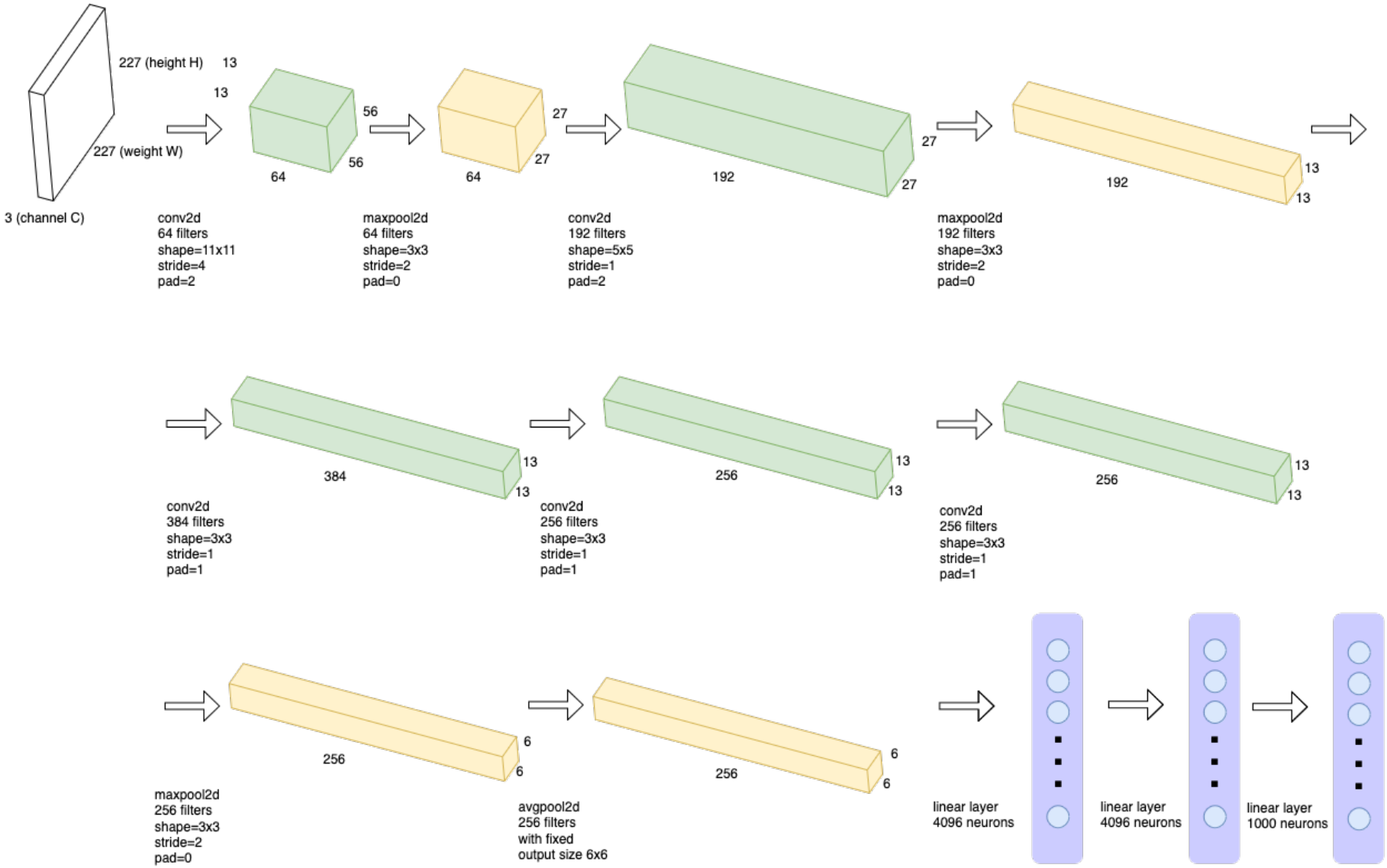






A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

AlexNet Network Architecture (PyTorch Library's implementation)



A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Popular CNN: AlexNet

```
[torch.Size([3, 227, 227])] INPUT
[torch.Size([64, 56, 56])] CONV1: 64 convolutional filters each with a shape=11x11, stride=4, pad=2
[torch.Size([64, 27, 27])] MAX POOL1: 3x3 max pooling filters with stride=2, pad=0
[torch.Size([192, 27, 27])] CONV2: 192 convolutional filters each with a shape=5x5, stride=1, pad=2
[torch.Size([192, 13, 13])] MAX POOL2: 3x3 max pooling filters with stride=2, pad=0
[torch.Size([384, 13, 13])] CONV3: 384 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([256, 13, 13])] CONV4: 256 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([256, 13, 13])] CONV5: 256 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([256, 6, 6])] MAX POOL3: 3x3 max pooling filters with stride=2, pad=0
[torch.Size([256, 6, 6])] AVG POOL: fixed output size [width x height]
[torch.Size([4096])] FC6: neurons
[torch.Size([4096])] FC7: neurons
[torch.Size([4])] FC8: neurons
```

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Group Activity: Inference with AlexNet

https://github.com/alimoorreza/cs195-fall24-notes/blob/main/cs195_alexnet_dissection.ipynb

A. Krizhevsky, I. Sutskever, and G. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS 2012

Popular CNN: VGG-16

- VGG-16 was the winner of ImageNet (1000-class image classification) challenge in 2014
  - proposed by Andrew Zisserman's group in Oxford University

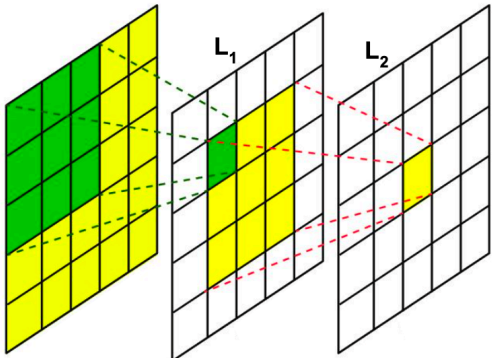
Karen Simonyan and Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2014

Popular CNN: VGG-16

- VGG-16's important features. VGG was the winner of ImageNet challenge in 2014:

- Bigger model compared to AlexNet
  - vgg16: 13 convolutional layers + 3 linear layers
  - vgg19: 16 convolutional layers + 3 linear layers
  - last 3 linear layers are the same as AlexNet
  - approximately 140,000,000 params

- Stacking 3x3 convolution filters:
  - fixed 3x3 convolution filters but many of them
  - within a sublayer, multiple 3x3 convolution filters are stacked together
    - stacking them together reduces the number of parameters needed to cover similar fields of view



Published as a conference paper at ICLR 2015

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

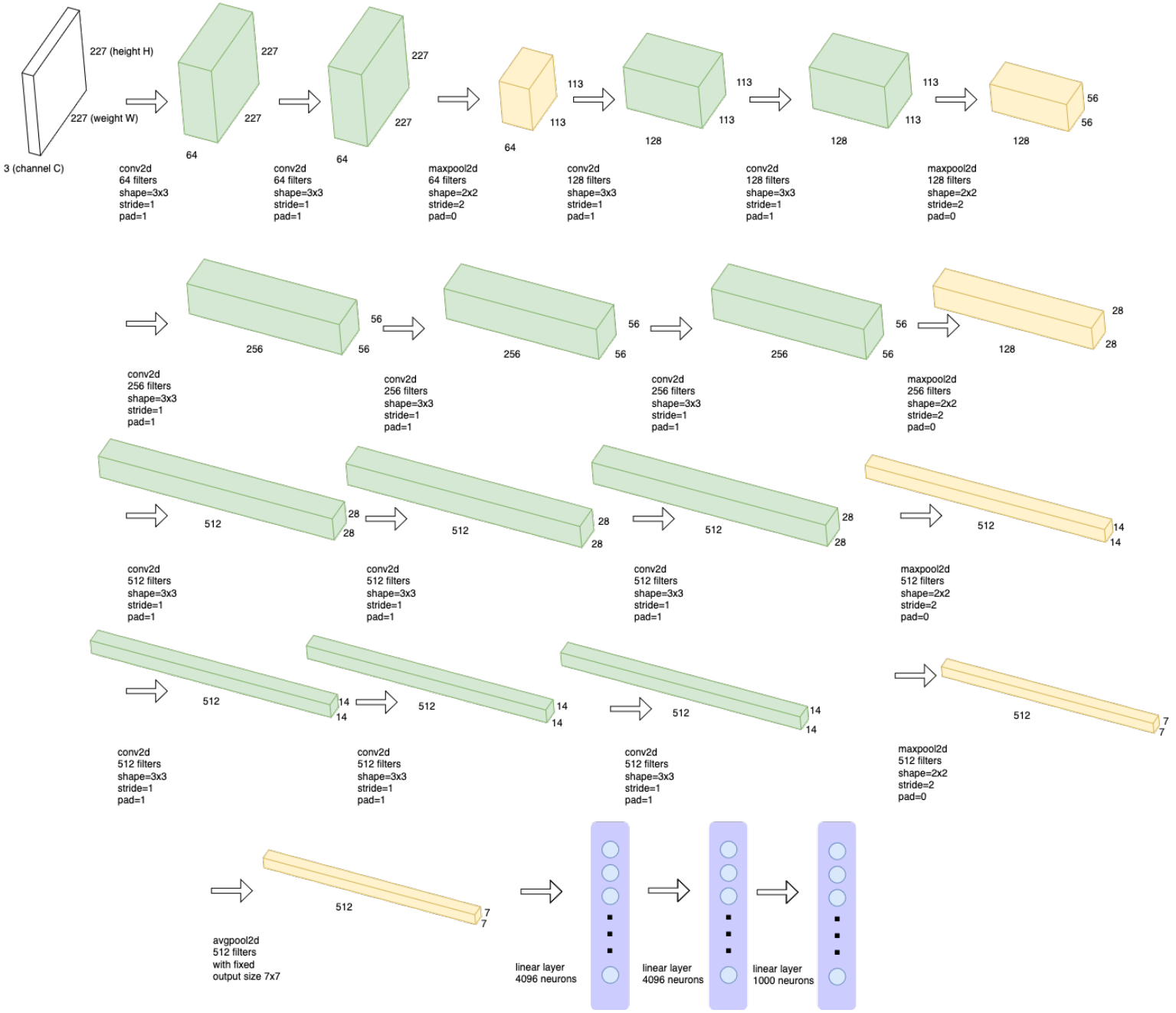
Karen Simonyan* **& Andrew Zisserman+**
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

ABSTRACT

In this work we investigate the effect of the convolutional network depth on its accuracy in the large-scale image recognition setting. Our main contribution is a thorough evaluation of networks of increasing depth using an architecture with very small (3×3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16–19 weight layers. These findings were the basis of our ImageNet Challenge 2014 submission, where our team secured the first and the second places in the localisation and classification tracks respectively. We also show that our representations generalise well to other datasets, where they achieve state-of-the-art results. We have made our two best-performing ConvNet models publicly available to facilitate further research on the use of deep visual representations in computer vision.

Karen Simonyan and Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

VGG-16 Network Architecture (PyTorch Library's implementation)



Popular CNN: VGG-16

```
[torch.Size([3, 227, 227])] INPUT
[torch.Size([64, 227, 227])] CONV1_1: 64 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([64, 227, 227])] CONV1_2: 64 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([64, 113, 113])] MAX POOL1: 2x2 max pooling filters with stride=2, pad=0
[torch.Size([128, 113, 113])] CONV2_1: 128 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([128, 113, 113])] CONV2_2: 128 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([128, 56, 56])] MAX POOL2: 2x2 max pooling filters with stride=2, pad=0
[torch.Size([256, 56, 56])] CONV3_1: 256 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([256, 56, 56])] CONV3_2: 256 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([256, 56, 56])] CONV3_3: 256 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([256, 28, 28])] MAX POOL3: 2x2 max pooling filters with stride=2, pad=0
[torch.Size([512, 28, 28])] CONV4_1: 512 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([512, 28, 28])] CONV4_2: 512 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([512, 28, 28])] CONV4_3: 512 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([512, 14, 14])] MAX POOL4: 2x2 max pooling filters with stride=2, pad=0
[torch.Size([512, 14, 14])] CONV5_1: 512 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([512, 14, 14])] CONV5_2: 512 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([512, 14, 14])] CONV5_3: 512 convolutional filters each with a shape=3x3, stride=1, pad=1
[torch.Size([512, 7, 7])] MAX POOL5: 2x2 max pooling filters with stride=2, pad=0
[torch.Size([512, 7, 7])] AVG POOL: fixed output size [width x height]
[torch.Size([4096])] FC6: neurons
[torch.Size([4096])] FC7: neurons
[torch.Size([1000])] FC8: neurons
```

VGG16

Karen Simonyan and Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2014

Group Activity: Inference with VGG-16

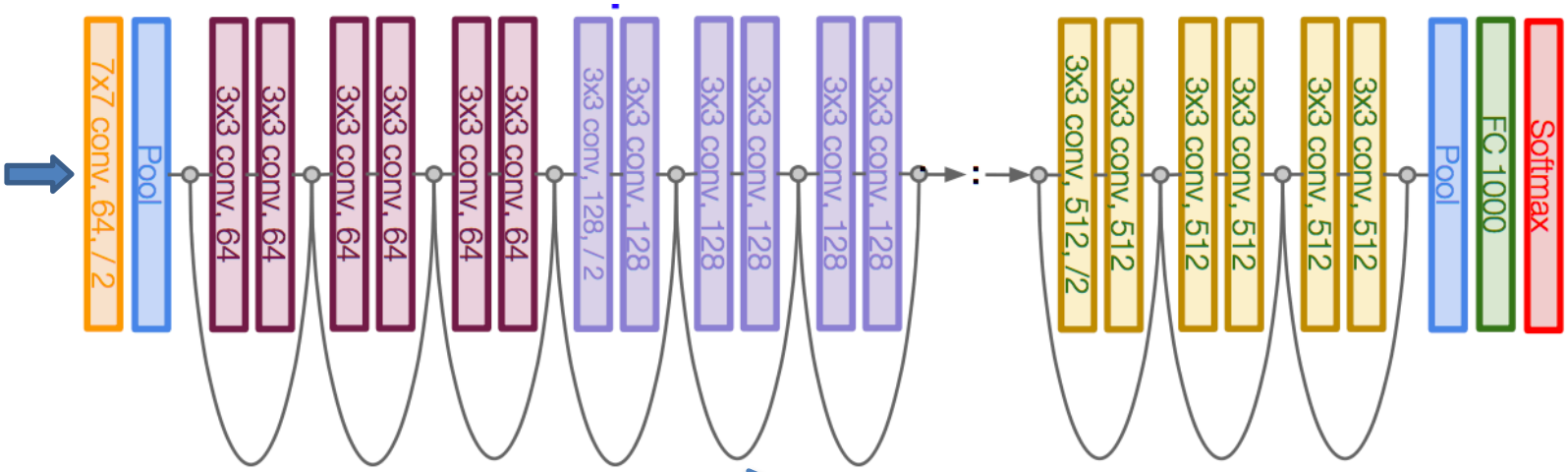
https://github.com/alimoorreza/cs195-fall24-notes/blob/main/cs195_vgg16_dissection.ipynb

Karen Simonyan and Andrew Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition, ICLR 2015

Popular CNN: ResNet

- ResNet was the winner of ImageNet challenge in 2015

Input image

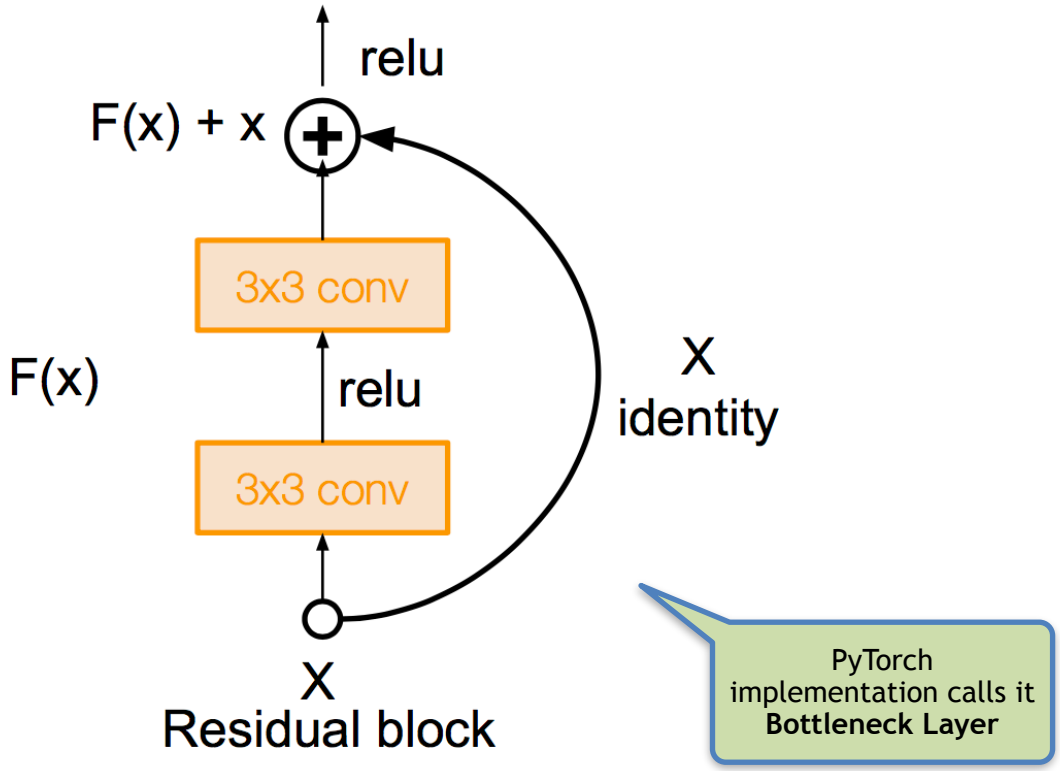


- Residual connections
- Skip connections

Many names but they refer to the same thing

Kaiming He et al., Deep Residual Learning for Image Recognition, CVPR 2015

ResNet's Main Innovation: Residual Block



Kaiming He et al., Deep Residual Learning for Image Recognition, CVPR 2015

ResNet's Main Innovation: Residual Block

Check out the notebook for details

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
	56×56	3×3 max pool, stride 2				
Layer1		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
Layer2	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
Layer3	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
Layer4	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Kaiming He et al., Deep Residual Learning for Image Recognition, CVPR 2015

ResNet: Network Architecture

```
class ResNet152(nn.Module):
    def __init__(self, num_classes, pretrained=True):
        super(ResNet152, self).__init__()
        if pretrained:
            net = models.resnet152(pretrained=True)

        num_features = net.fc.in_features
        net.fc = nn.Linear(num_features, num_classes)

        self.conv1 = net.conv1
        self.bn1 = net.bn1
        self.relu = net.relu
        self.maxpool = net.maxpool

        self.layer1 = net.layer1
        self.layer2 = net.layer2
        self.layer3 = net.layer3
        self.layer4 = net.layer4

        self.avgpool = net.avgpool
        self.fc = net.fc
        #pdb.set_trace()

    def forward(self, x):
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu(x)
        x = self.maxpool(x)

        x = self.layer1(x)
        x = self.layer2(x)
        x = self.layer3(x)
        x = self.layer4(x)

        x = self.avgpool(x)
        x = torch.flatten(x, 1)
        x = self.fc(x)

        return x
```

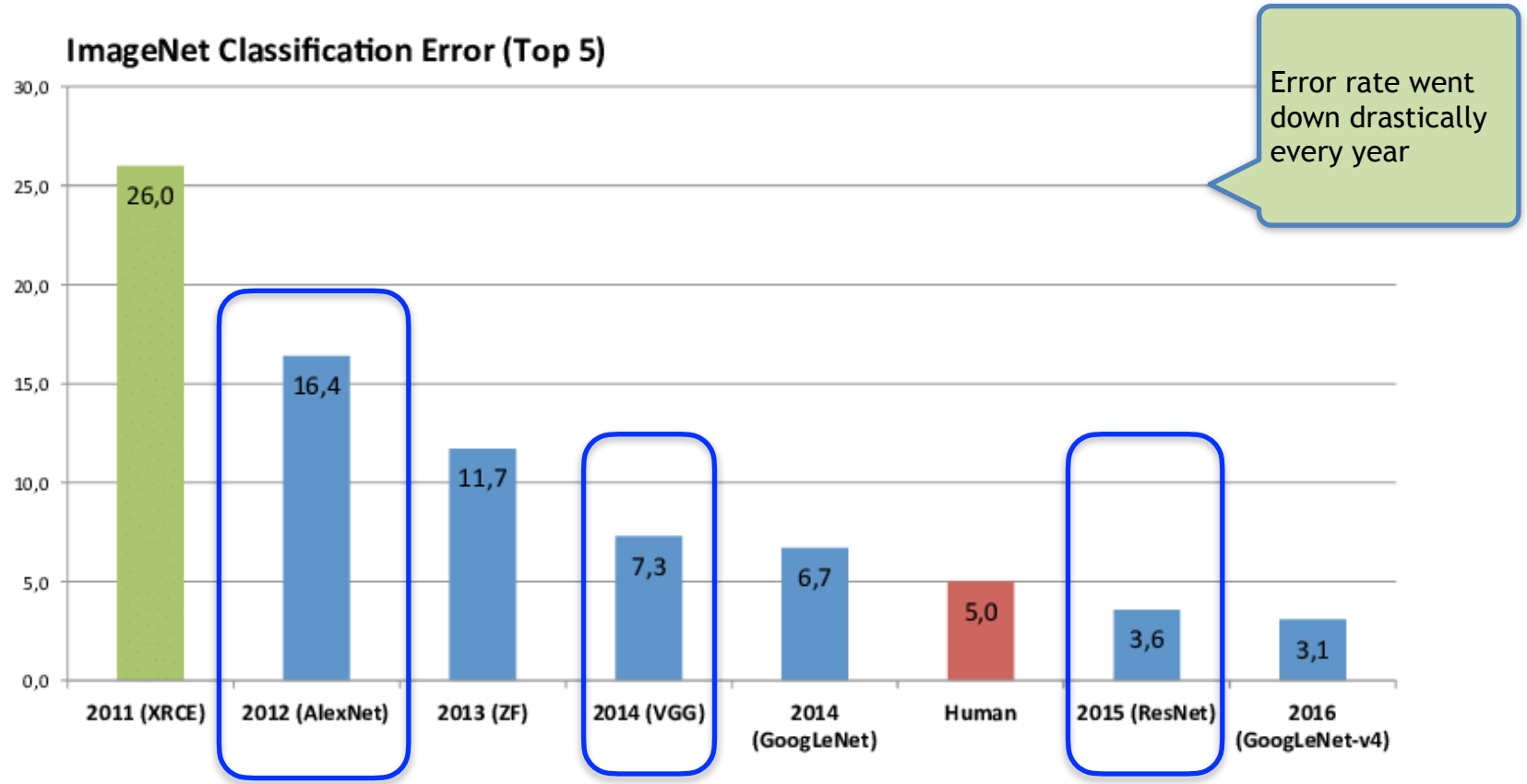
Group Activity: Inference with ResNet

https://github.com/alimoorreza/cs195-fall24-notes/blob/main/cs195_resnet_dissection.ipynb

Kaiming He et al., Deep Residual Learning for Image Recognition, CVPR 2015

ImageNet Winners by the Popular CNNs

- AlexNet (2012) —> VGG (2014)—> ResNet (2015)



Popular Model Implementation on Huggingface

List of models from huggingface

Models

All model architecture families include variants with pretrained weights. There are specific model variants without any weights, it is NOT a bug. Help training new or better weights is always appreciated.

- Aggregating Nested Transformers - https://arxiv.org/abs/2105.12723
- BEiT - https://arxiv.org/abs/2106.08254
- Big Transfer ResNetV2 (BiT) - https://arxiv.org/abs/1912.11370
- Bottleneck Transformers - https://arxiv.org/abs/2101.11605
- CaiT (Class-Attention in Image Transformers) - https://arxiv.org/abs/2103.17239
- CoaT (Co-Scale Conv-Attentional Image Transformers) - https://arxiv.org/abs/2104.06399
- CoAtNet (Convolution and Attention) - https://arxiv.org/abs/2106.04803
- ConvNeXt - https://arxiv.org/abs/2201.03545
- ConvNeXt-V2 - http://arxiv.org/abs/2301.00808
- ConViT (Soft Convolutional Inductive Biases Vision Transformers)- https://arxiv.org/abs/2103.10697
- CspNet (Cross-Stage Partial Networks) - https://arxiv.org/abs/1911.11929
- DeiT - https://arxiv.org/abs/2012.12877
- DeiT-III - https://arxiv.org/pdf/2204.07118.pdf
- DenseNet - https://arxiv.org/abs/1608.06993
- DLA - https://arxiv.org/abs/1707.06484
- DPN (Dual-Path Network) - https://arxiv.org/abs/1707.01629
- EdgeNeXt - https://arxiv.org/abs/2206.10589
- EfficientFormer - https://arxiv.org/abs/2206.01191
- EfficientNet (MBConvNet Family)