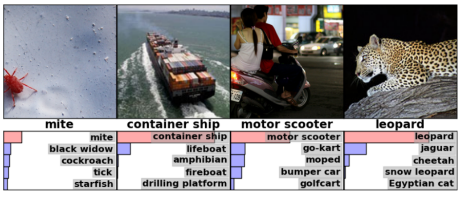
Convolutional Neural Net

Alex Olson

Introduction: Why ConvNets

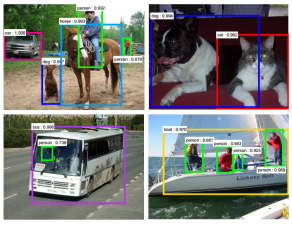
- **Fully Connected Networks Challenges:**
  - High dimensionality of inputs
  - Loss of spatial structure when flattening inputs
- **Convolutional Neural Networks (CNNs):**
  - Address high-dimensional input challenges
  - Preserve spatial structure through local receptive
    parameter sharing
  - Efficient for pattern recognition due to reduced pa
    and computational efficiency

Used everywhere for Vision



[Krizhevsky 2012]

[Ciresan et al. 2013]



[Faster R-CNN - Ren 2015]

[NVIDIA dev blog]

Many other applications

Speech recognition & speech synthesis

Many other applications

Speech recognition & speech synthesis

Natural Language Processing

Many other applications

Speech recognition & speech synthesis

Natural Language Processing

Protein/DNA binding prediction

Many other applications

Speech recognition & speech synthesis

Natural Language Processing

Protein/DNA binding prediction

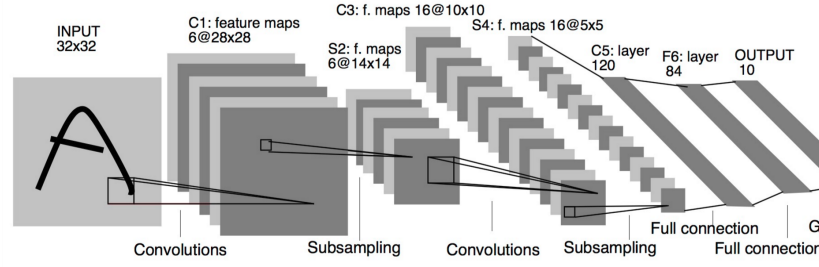
Any problem with a spatial (or sequential) str

ConvNets for image classification

CNN = Convolutional Neural Networks = ConvNet

ConvNets for image classification

CNN = Convolutional Neural Networks = ConvNet



LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning app

Outline

Convolutions

Outline

Convolutions

CNNs for Image Classification

Outline

Convolutions

CNNs for Image Classification

CNN Architectures

Convolutions

Motivations

Standard Dense Layer for an image input:

```
x = Input((640, 480, 3), dtype='float32')
# shape of x is: (None, 640, 480, 3)
x = Flatten()(x)
# shape of x is: (None, 640 x 480 x 3)
z = Dense(1000)(x)
```

How many parameters in the Dense layer?

Motivations

Standard Dense Layer for an image input:

```
x = Input((640, 480, 3), dtype='float32')
# shape of x is: (None, 640, 480, 3)
x = Flatten()(x)
# shape of x is: (None, 640 x 480 x 3)
z = Dense(1000)(x)
```

How many parameters in the Dense layer?

$640 \times 480 \times 3 \times 1000 + 1000 = 922M$!

Motivations

Standard Dense Layer for an image input:

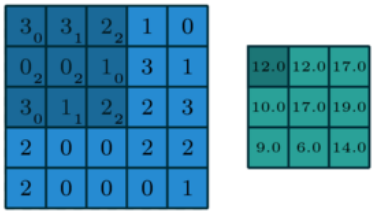
```
x = Input((640, 480, 3), dtype='float32')
# shape of x is: (None, 640, 480, 3)
x = Flatten()(x)
# shape of x is: (None, 640 x 480 x 3)
z = Dense(1000)(x)
```

How many parameters in the Dense layer?

$640 \times 480 \times 3 \times 1000 + 1000 = 922M$!

Spatial organization of the input is destroyed by Flatten

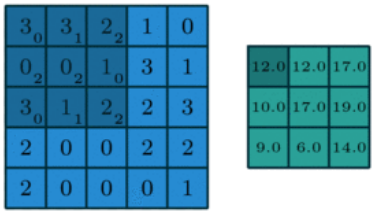
Convolution in a neural network



- x is a 3×3 chunk (dark area) of the image *(blue array)*
- Each output neuron is parametrized with the 3×3 weight mat
  numbers)

These slides extensively use convolution visualisation by V. Dumoulin available at [ht
conv_arithmetic]

Convolution in a neural network



- x is a 3×3 chunk (dark area) of the image *(blue array)*
- Each output neuron is parametrized with the 3×3 weight mat
  numbers)

The activation obtained by sliding the 3×3 window and computir

$$z(x) = relu(\mathbf{w}^T x + b)$$

Motivations

Local connectivity

- A neuron depends only on a few local input neurons
- Translation invariance

Motivations

Local connectivity

- A neuron depends only on a few local input neurons
- Translation invariance

Comparison to Fully connected

- Parameter sharing: reduce overfitting
- Make use of spatial structure: **strong prior** for vision!

Motivations

Local connectivity

- A neuron depends only on a few local input neurons
- Translation invariance

Comparison to Fully connected

- Parameter sharing: reduce overfitting
- Make use of spatial structure: **strong prior** for vision!

Animal Vision Analogy

Hubel & Wiesel, RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORT

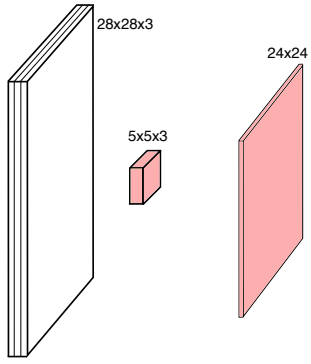
Channels

Colored image = tensor of shape (height, width, channels)

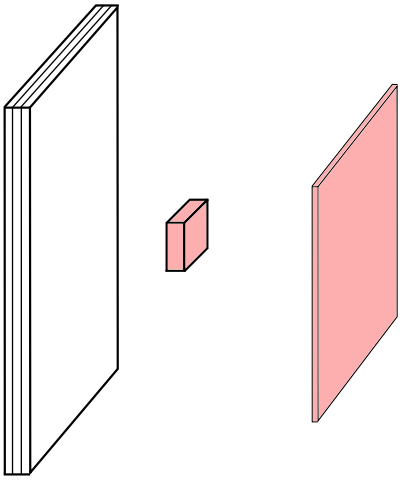
Channels

Colored image = tensor of shape (height, width, channels)

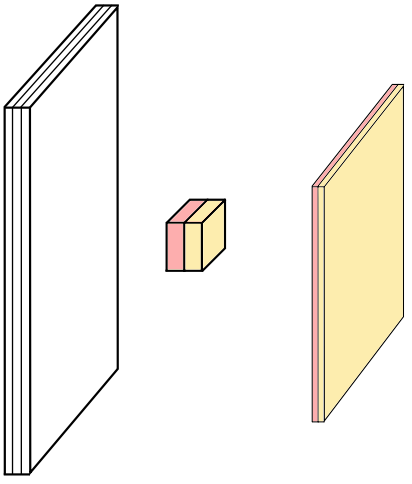
Convolutions are usually computed for each channel and summed:



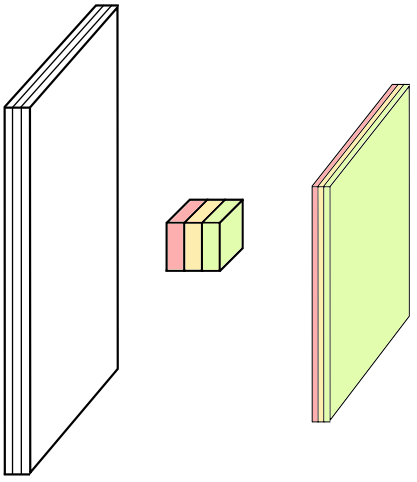
Multiple convolutions



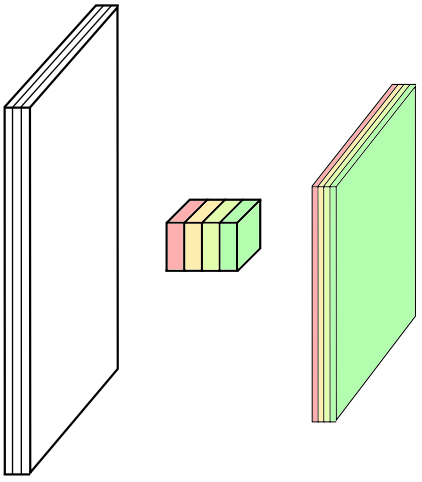
Multiple convolutions



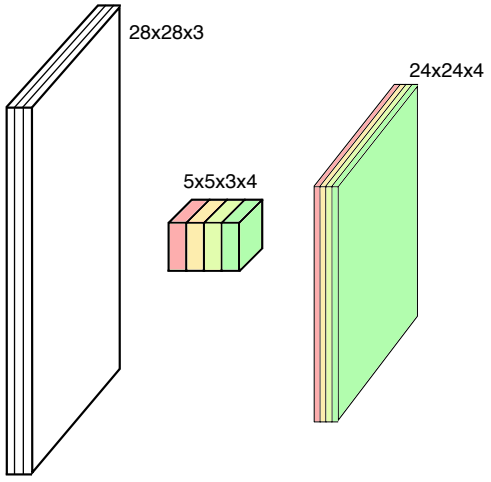
Multiple convolutions



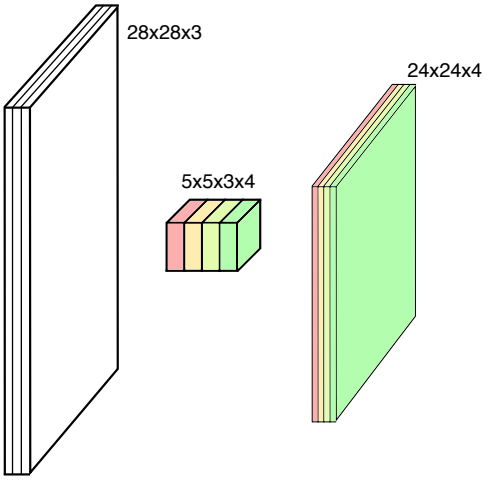
Multiple convolutions



Multiple convolutions



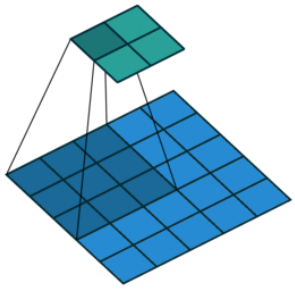
Multiple convolutions



- Kernel size aka receptive field (usually 1, 3, 5, 7, 11)
- Output dimension: length - kernel_size + 1

Strides

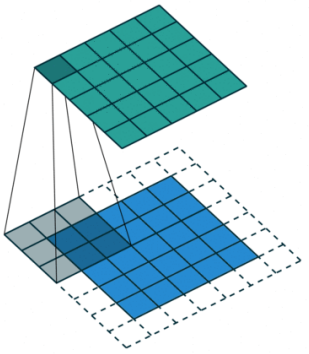
- Strides: increment step size for the convolution operator
- Reduces the size of the output map



Example with kernel size 3×3 and a stride of 2 (image in blu

Padding

- Padding: artificially fill borders of image
- Useful to keep spatial dimension constant across filters
- Useful with strides and large receptive fields
- Usually: fill with 0s

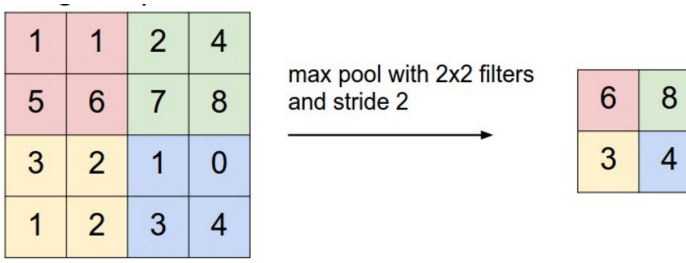


Pooling

- Spatial dimension reduction
- Local invariance
- No parameters: max or average of 2x2 units

Pooling

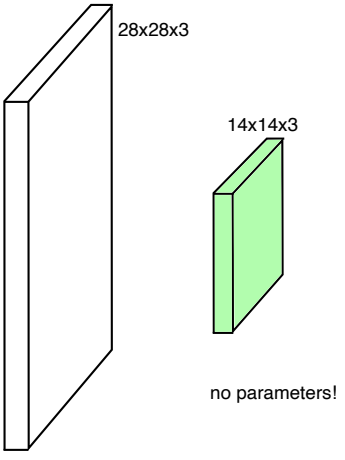
- Spatial dimension reduction
- Local invariance
- No parameters: max or average of 2x2 units



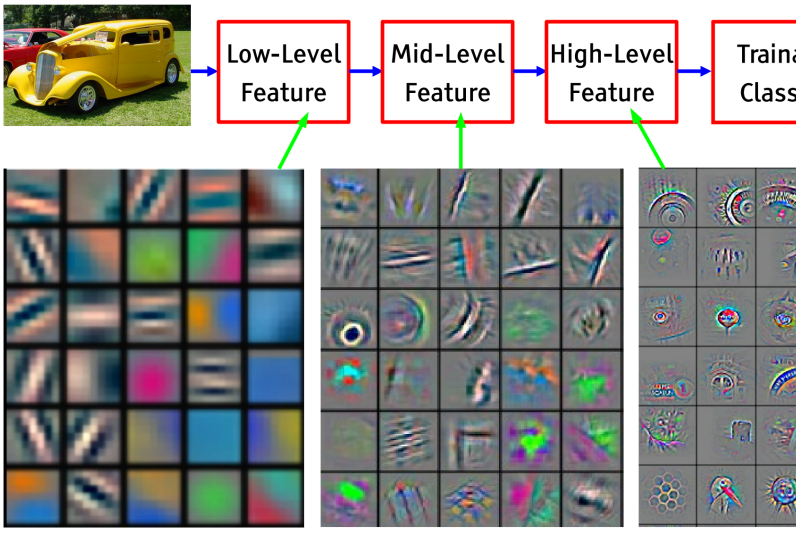
Schematic from Stanford http://cs231n.github.io/convolutional-networks

Pooling

- Spatial dimension reduction
- Local invariance
- No parameters: max or average of 2x2 units

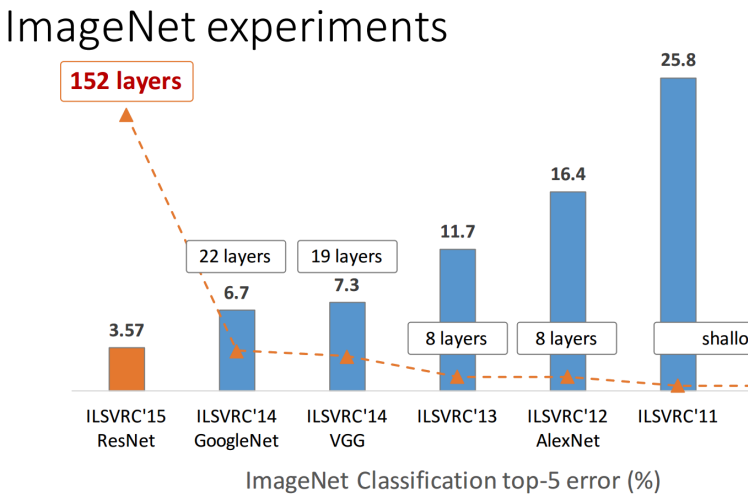


Hierarchical representation



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

Deeper is better



from Kaiming He slides "Deep residual learning for image recognition." ICML. 2016.

Pre-trained models

Pre-trained models

Training a model on ImageNet from scratch takes **days or weeks**.

Pre-trained models

Training a model on ImageNet from scratch takes **days or weeks**.

Many models trained on ImageNet and their weights are publicly a

Pre-trained models

Training a model on ImageNet from scratch takes **days or weeks**.

Many models trained on ImageNet and their weights are publicly a

Transfer learning

- Use pre-trained weights, remove last layers to compute represe
- Train a classification model from these features on a new classi
- The network is used as a generic feature extractor
- Better than handcrafted feature extraction on natural images

Pre-trained models

Training a model on ImageNet from scratch takes **days or weeks**.

Many models trained on ImageNet and their weights are publicly a

Fine-tuning

Retraining the (some) parameters of the network (given enough data)

Pre-trained models

Training a model on ImageNet from scratch takes **days or weeks**.

Many models trained on ImageNet and their weights are publicly a

Fine-tuning

Retraining the (some) parameters of the network (given enough data)

- Truncate the last layer(s) of the pre-trained network
- Freeze the remaining layers weights
- Add a (linear) classifier on top and train it for a few epochs

Pre-trained models

Training a model on ImageNet from scratch takes **days or weeks**.

Many models trained on ImageNet and their weights are publicly a

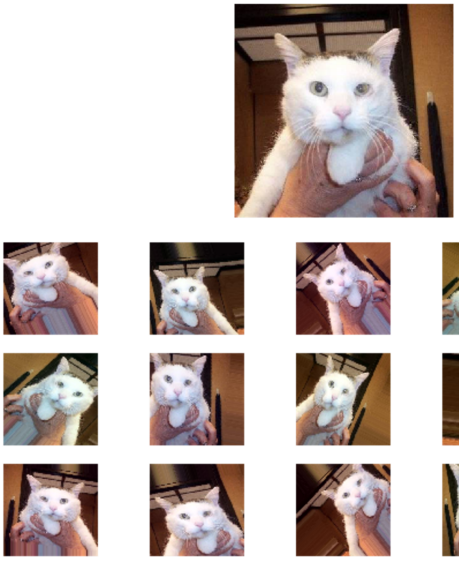
Fine-tuning

Retraining the (some) parameters of the network (given enough data)

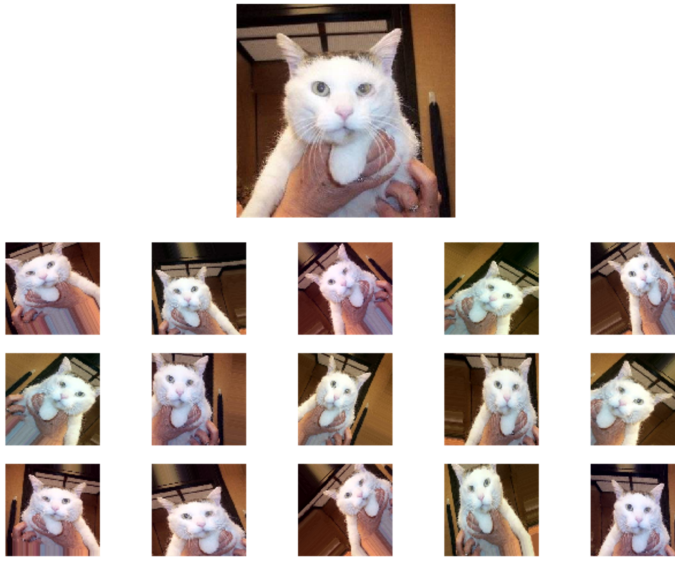
- Truncate the last layer(s) of the pre-trained network
- Freeze the remaining layers weights
- Add a (linear) classifier on top and train it for a few epochs
- Then fine-tune the whole network or the few deepest layers
- Use a smaller learning rate when fine tuning

Data Augmentation

Data Augmentation



Data Augmentation



See also: RandAugment and Unsupervised Data Augmentation for

Next: Lab