COMSM0045: Convolutional Neural Networks (Part 2)

Dima Damen
Dima.Damen@bristol.ac.uk

Bristol University, Department of Computer Science
Bristol BS8 1UB, UK

October 11, 2020

Convolutional Neural Networks

- And now... to the main attraction **Convolutional Neural Networks (CNN)**

Convolutional Neural Networks

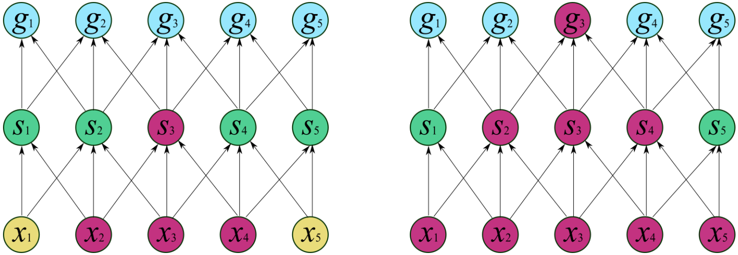
- And now... to the main attraction **Convolutional Neural Networks (CNN)**
- Three primary properties distinguish fully-connected networks from convolutional neural networks:

Convolutional Neural Networks

- And now... to the main attraction **Convolutional Neural Networks (CNN)**
- Three primary properties distinguish fully-connected networks from convolutional neural networks:
    1. Sparse Interactions
    2. Parameter Sharing
    3. Equi-variant Representations

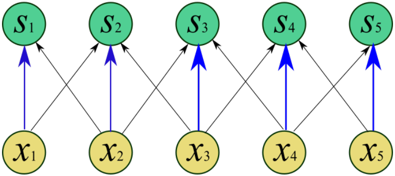
CNN Properties: 1- Sparse Interactions

- **The receptive field of the units in the deeper layers of a CNN is larger than the receptive field of the units in the shallow layers**



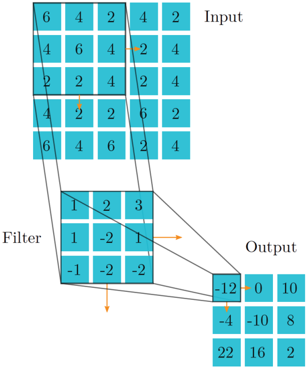
Dima Damen
Dima.Damen@bristol.ac.uk

CNN Properties: 2- Parameter Sharing



CNN Properties: 2- Parameter Sharing

- And in 2-D



Source: BSc Thesis, Will Price, Univ of Bristol, May 2017

CNN Properties: 3- Equi-variant Representations

CNN Properties: 3- Equi-variant Representations

- As a result of the first two properties, CNNs exhibit some equivariance properties.

CNN Properties: 3- Equi-variant Representations

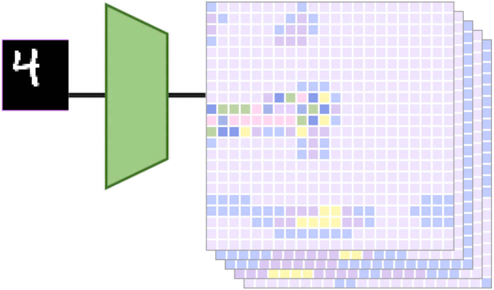
- As a result of the first two properties, CNNs exhibit some equivariance properties.
- A function is equivariant if when the input changes (or shifts) in a certain way, the output also changes in exactly the same way.

CNN Properties: 3- Equi-variant Representations

- As a result of the first two properties, CNNs exhibit some equivariance properties.
- A function is equivariant if when the input changes (or shifts) in a certain way, the output also changes in exactly the same way.
- CNNs are equi-variant to... translation

CNN Properties: 3- Equi-variant Representations

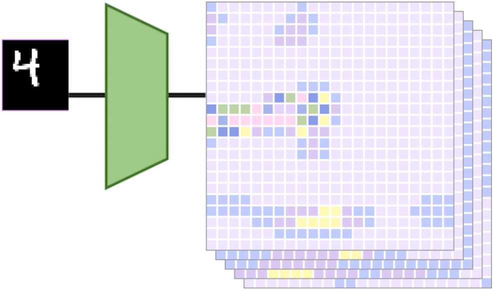
- CNNs are equi-variant to... translation
- This is of immense value in images for example.



Source: Christian Wolf Blog (2020) - What is translation equivariance, and why do we use convolutions to get it? https://link.medium.com/4u44mjdXqab

CNN Properties: 3- Equi-variant Representations

- CNNs are equi-variant to... translation
- This is of immense value in images for example.

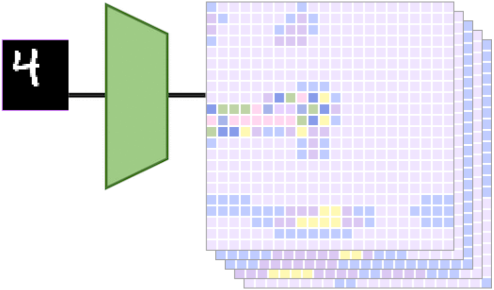


- If an object moves in the input, its representation will move in the output in the same direction.

Source: Christian Wolf Blog (2020) - What is translation equivariance, and why do we use convolutions to get it? https://link.medium.com/4u44mjdXqab

CNN Properties: 3- Equi-variant Representations

- CNNs are equi-variant to... translation
- This is of immense value in images for example.

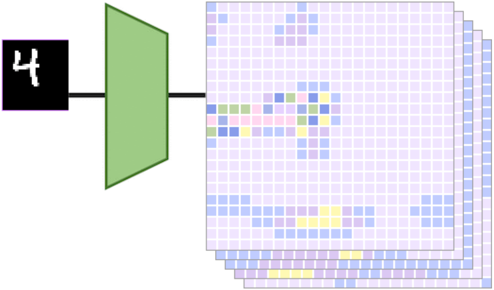


- If an object moves in the input, its representation will move in the output in the same direction.
- However CNNs are NOT equivariant to...

Source: Christian Wolf Blog (2020) - What is translation equivariance, and why do we use convolutions to get it? https://link.medium.com/4u44mjdXqab

CNN Properties: 3- Equi-variant Representations

- CNNs are equi-variant to... translation
- This is of immense value in images for example.



- If an object moves in the input, its representation will move in the output in the same direction.
- However CNNs are NOT equivariant to... rotation or scale

Source: Christian Wolf Blog (2020) - What is translation equivariance, and why do we use convolutions to get it? https://link.medium.com/4u44mjdXqab

Zero Padding

- However, to make the most of the input, particularly around the edges/borders, one essential feature of any CNN implementation is **zero padding** the input to make it wider
- Without zero-padding, the input shrinks by one pixel less than the kernel width at each layer
- With zero-padding, the input and output are of the same size, unlike example below

Zero Padding

- However, to make the most of the input, particularly around the edges/borders, one essential feature of any CNN implementation is **zero padding** the input to make it wider
- Without zero-padding, the input shrinks by one pixel less than the kernel width at each layer
- With zero-padding, the input and output are of the same size, unlike example below
- Without zero-padding, the number of convolutional layers that can be included in a network will be capped

CNN Architecture Considerations

- In images for example, we have 3 channels (R/G/B)
- This means the input is 3D, and thus our convolutions are necessarily 3-D tensors

CNN Architecture Considerations

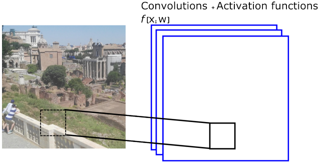
- In images for example, we have 3 channels (R/G/B)
- This means the input is 3D, and thus our convolutions are necessarily 3-D tensors
- Moreover, these are run in batch mode, so are typically 4-D tensors, with the fourth dimension indexing different examples in the batch

CNN Architecture Considerations

- In images for example, we have 3 channels (R/G/B)
- This means the input is 3D, and thus our convolutions are necessarily 3-D tensors
- Moreover, these are run in batch mode, so are typically 4-D tensors, with the fourth dimension indexing different examples in the batch
- All CNN representations omit the fourth dimension, for batch-based optimisation, for simplicity

CNN Architecture Considerations

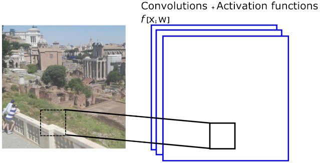
- In the previous example, the input and output sizes of the convolution+activation layers were equal



- However, practically we do not convolve densely, but instead we move the convolution skipping certain pixels.

CNN Architecture Considerations

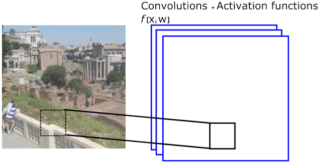
- In the previous example, the input and output sizes of the convolution+activation layers were equal



- However, practically we do not convolve densely, but instead we move the convolution skipping certain pixels.
- The number of pixels we skip, is referred to as the **stride** of the layer

CNN Architecture Considerations

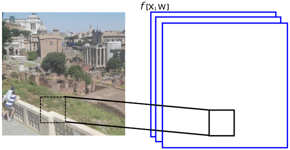
- In the previous example, the input and output sizes of the convolution+activation layers were equal



- However, practically we do not convolve densely, but instead we move the convolution skipping certain pixels.
- The number of pixels we skip, is referred to as the **stride** of the layer
- This results in downsampled convolutions

CNN Architecture Considerations

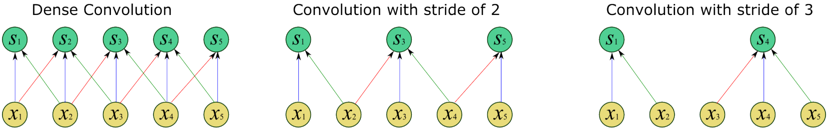
- In the previous example, the input and output sizes of the convolution+activation layers were equal



- However, practically we do not convolve densely, but instead we move the convolution skipping certain pixels.
- The number of pixels we skip, is referred to as the **stride** of the layer
- This results in downsampled convolutions
- It is possible to use separate strides for each dimension

CNN Architecture Considerations

- An example of coonvolution with a stride of two, and a stride of three, is shown below



CNN Architecture Considerations

- Care should be taken when backpropagating CNNs with zero padding or stride of more than one.

CNN Architectures - AlexNet

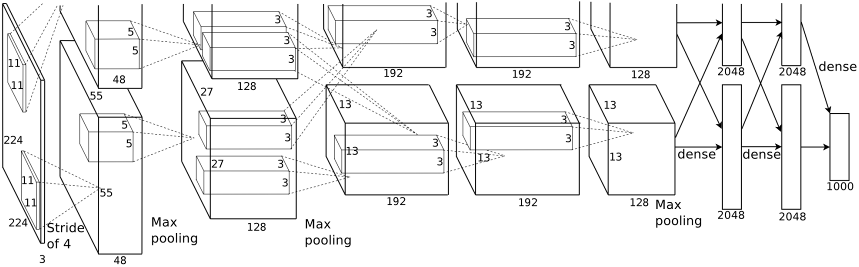
- When AlexNet won the most challenging computer vision task - Classifying 1000 classes by training from 10,000,000 images (The ImageNet Challenge), the new wave of CNN architectures started

---

Alex Krizhevsky, Sutskever and Hinton (2012) ImageNet Classification with Deep Convolutional Neural Networks, NIPS

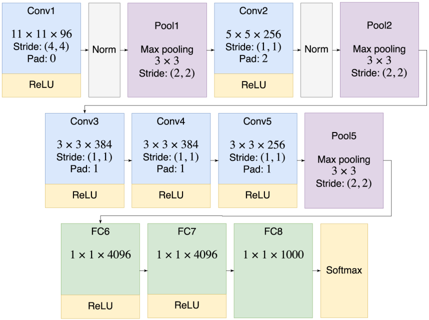
CNN Architectures - AlexNet

- When AlexNet won the most challenging computer vision task - Classifying 1000 classes by training from 10,000,000 images (The ImageNet Challenge), the new wave of CNN architectures started



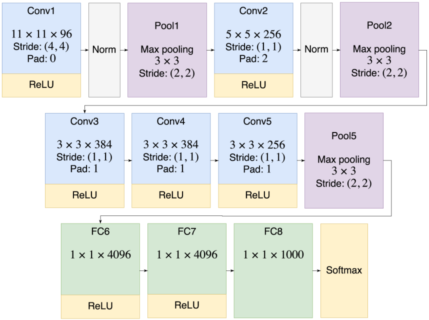
Alex Krizhevsky, Sutskever and Hinton (2012) ImageNet Classification with Deep Convolutional Neural Networks, NIPS

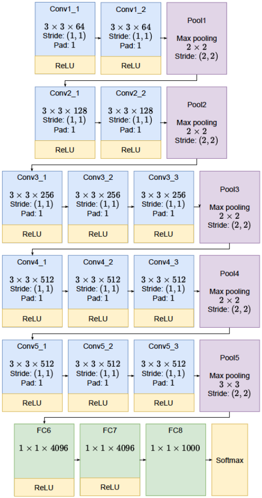
CNN Architectures - AlexNet vs VGG-16



Source: BSc Thesis, Will Price, Univ of Bristol, May 2017

CNN Architectures - AlexNet vs VGG-16





Source: BSc Thesis, Will Price, Univ of Bristol, May 2017

Training CNNs

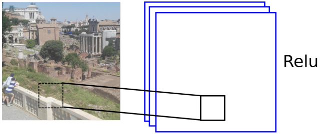
- The most expensive part of CNN training is learning the features
- The fully-connected layers are usually relatively inexpensive to train because of the small number of features provided as input

Training CNNs

- The most expensive part of CNN training is learning the features
- The fully-connected layers are usually relatively inexpensive to train because of the small number of features provided as input
- When performing gradient descent, every gradient step requires a complete run of forward propagation and backward propagation through the entire network

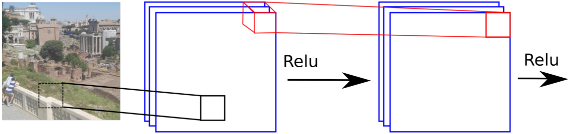
Recent Architectures - Residual Networks (ResNet)

- Finding the right weights for a deep convolutional network is not easy.
- A trick introduced by He et al (2015) suggested adding *shortcuts* in the network architecture



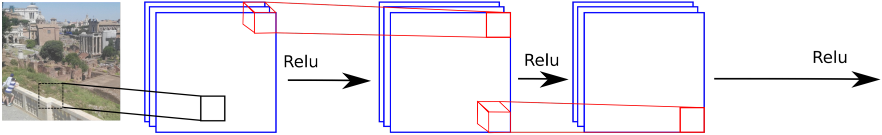
Recent Architectures - Residual Networks (ResNet)

- Finding the right weights for a deep convolutional network is not easy.
- A trick introduced by He et al (2015) suggested adding *shortcuts* in the network architecture



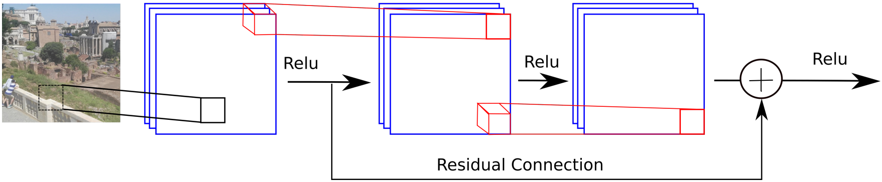
Recent Architectures - Residual Networks (ResNet)

- Finding the right weights for a deep convolutional network is not easy.
- A trick introduced by He et al (2015) suggested adding *shortcuts* in the network architecture



Recent Architectures - Residual Networks (ResNet)

- Finding the right weights for a deep convolutional network is not easy.
- A trick introduced by He et al (2015) suggested adding *shortcuts* in the network architecture



Recent Architectures - Residual Networks (ResNet)

- ResNet block

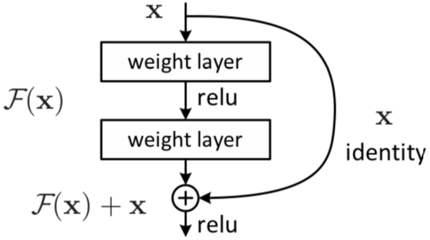


Figure 2. Residual learning: a building block.

- Allowed faster convergence - by searching for weights that deviate slightly from the identity

Source: He et al 2015. Deep Residual Learning for Image Recognition

Recent Architectures - Residual Networks (ResNet)

- ResNet block

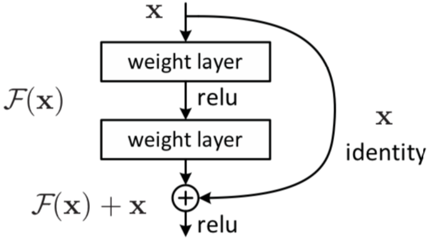


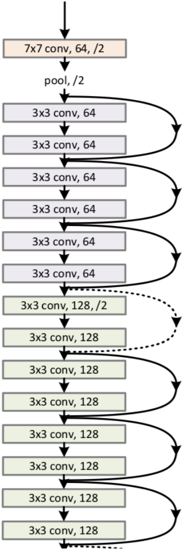
Figure 2. Residual learning: a building block.

- Allowed faster convergence - by searching for weights that deviate slightly from the identity
- Allowed training deeper networks - ResNet-50, ResNet-101

Source: He et al 2015. Deep Residual Learning for Image Recognition

Recent Architectures - Residual Networks (ResNet)

- Part of a ResNet



Source: He et al 2015. Deep Residual Learning for Image Recognition

The latest!

- ResNext blocks

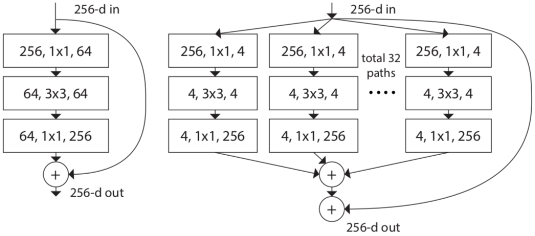


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

Source: Xie et al 2017. Aggregated Residual Transformations for Deep Neural Networks

CNN Architectures - Further architectures

- See live demos at: `http://cs231n.stanford.edu`
- Visualise recent architectures at: `http://josephpcohen.com/w/visualizing-cnn-architectures-side-by-side-with-mxnet/`

Dima Damen
Dima.Damen@bristol.ac.uk

Further Reading

- Deep Learning

  Ian Goodfellow, Yoshua Bengio, and Aaron Courville
  MIT Press, ISBN: 9780262035613.
  - Chapter 9 – Convolutional Networks