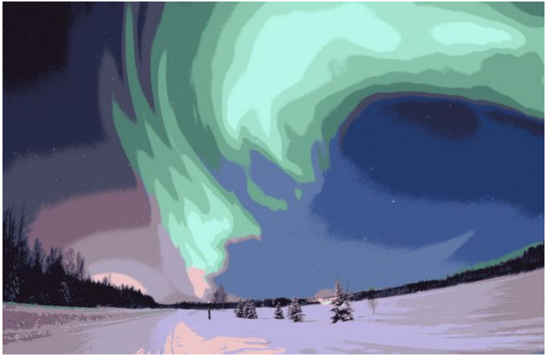
Image Segmentation

Image Segmentation

- Segmentation in Humans: human visual system's ability to distinguish objects in a scene.
- Segmentation as Clustering: Clusters pixels based on similarity in color, intensity, or texture.
- **k-Means Segmentation:**
    - Centroid-Based Clustering: Assigns pixels to the nearest cluster center based on feature similarity.
    - Iterative Optimization: Continuously refines clusters to minimize within-cluster variance.

Image Segmentation

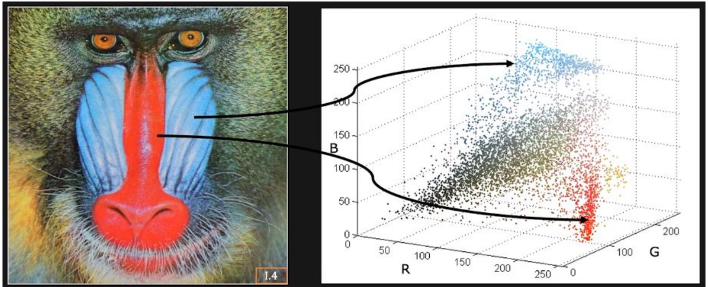


Pixels as feature vectors

Let i and j be two pixels whose features are $\mathbf{f}_i$ and $\mathbf{f}_j$.

$\mathcal{L}^2$ Distance between $\mathbf{f}_i$ and $\mathbf{f}_j$:

$$S(\mathbf{f}_i, \mathbf{f}_j) = \sqrt{\sum_k (f_{ik} - f_{jk})^2}$$



Input Image

Pixel RGB Color Distribution

(Color of feature point ≡ Color of image pixel)

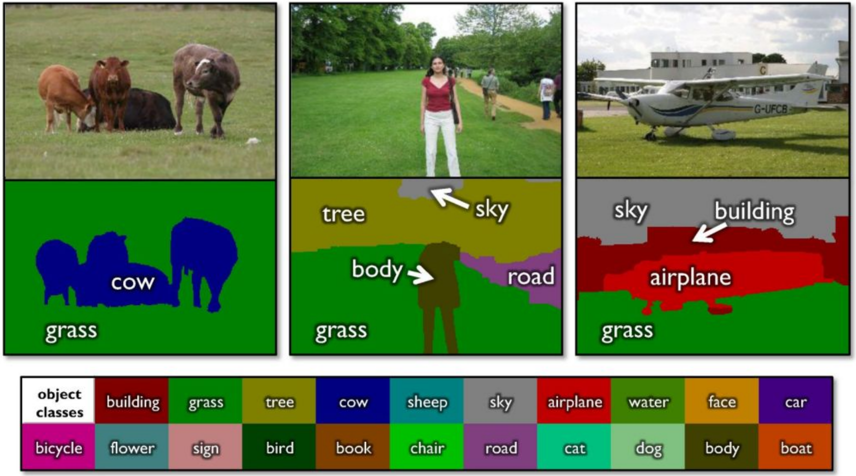
Normalized Cut for Image Segmentation

- **Graph Representation:** Image modeled as an undirected graph $G = (V, E)$, with vertices V and edges E.
- **Edge Weights:** Weights $w(v_i, v_j)$ on edges denote the similarity between pixels.
- **Cut Cost:** Cut between sets A and B: $\mathrm{cut}(A, B) = \sum_{u \in A, v \in B} w(u, v)$.
- **Partitioning Objective:** Minimize Normalized Cut value:
  - $\mathrm{Ncut}(A, B) = \frac{\mathrm{cut}(A,B)}{\mathrm{assoc}(A,V)} + \frac{\mathrm{cut}(A,B)}{\mathrm{assoc}(B,V)}$.
- **Association Metrics:** Association to graph:
  - $\mathrm{assoc}(A, V) = \sum_{u \in A, t \in V} w(u, t)$, and similarly for B.
- **Spectral Clustering Approach:** Use eigenvalues and eigenvectors of Laplacian $L = D - W$ to find partitions. Partition found via eigenvector for the second smallest eigenvalue of L.

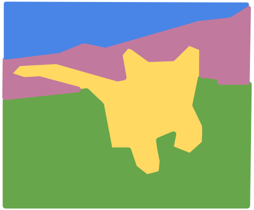
Image Segmentation

- Classical methods inspired the initial deep learning architectures for segmentation.
- **State-of-the-Art Techniques:**
  - Deep learning methods, especially CNNs, lead today's segmentation advancements.
  - Techniques like FCNs, U-Net, and DeepLab offer precise object boundary delineation.
- **Impact and Applications:**
  - Significant improvements in accuracy and detail over classical methods.
  - Wide-ranging applications from medical imaging to autonomous driving.

Semantic Segmentation



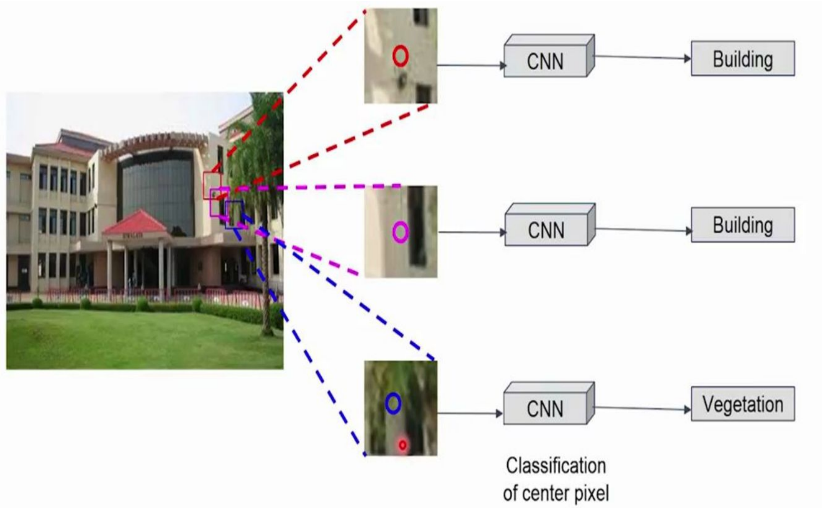
Semantic Segmentation



- Task of grouping together similar (in semantic content) pixels in an image. How to formulate this using DNNs? **Cast as a pixel classification problem!**
- For each training image, each pixel is labeled with a semantic category. Quite annotation-intensive!

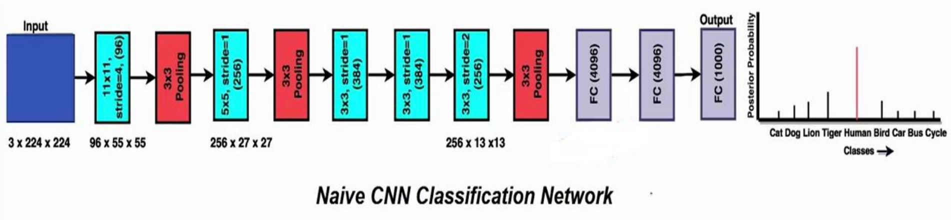
Credit: Fei-Fei Li et al, CS231n, Stanford Univ

Semantic Segmentation



Fully Convolutional Networks for Semantic Segmentation (FCN)[1]

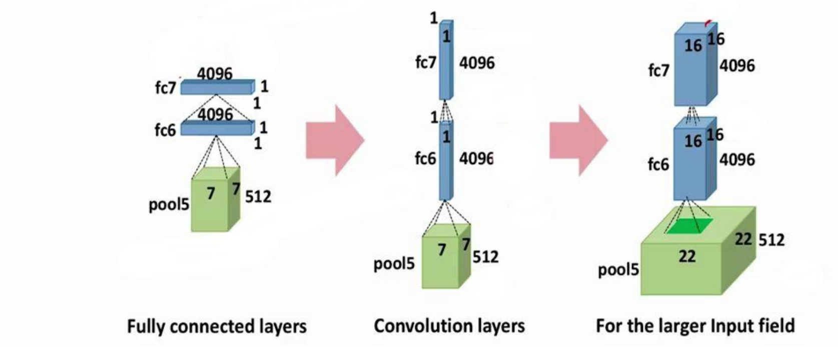
- Adapts various classification networks (VGG net, GoogLeNet) into fully convolutional networks by converting **FC layers into 1 × 1 conv layers**



[1]Shelhamer et al, Fully Convolutional Networks for Semantic Segmentation, TPAMI 2016

Fully Convolutional Networks for Semantic Segmentation (FCN)[1]

- To obtain classification for each pixel, another 1×1 conv layer is appended with channel dimension $C + 1$ where C is number of classes. Do you see any problem?



[1]Shelhamer et al, Fully Convolutional Networks for Semantic Segmentation, TPAMI 2016

Fully Convolutional Networks for Semantic Segmentation (FCN)[1]

- Image classification architectures perform downsampling as they go deeper $\Rightarrow$ fully convolutional architecture will have lower resolution than input. What to do?

[1] Shelhamer et al, Fully Convolutional Networks for Semantic Segmentation, TPAMI 2016

Fully Convolutional Networks for Semantic Segmentation (FCN)[1]

- Image classification architectures perform downsampling as they go deeper ⇒ fully convolutional architecture will have lower resolution than input. What to do?
- Perform upsampling to get back to original resolution. How to do?

[1] Shelhamer et al, Fully Convolutional Networks for Semantic Segmentation, TPAMI 2016

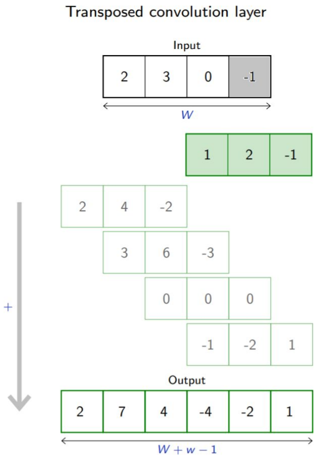
Fully Convolutional Networks for Semantic Segmentation (FCN)[1]

- Image classification architectures perform downsampling as they go deeper $\Rightarrow$ fully convolutional architecture will have lower resolution than input. What to do?
- Perform upsampling to get back to original resolution. Learnable upsampling done through **Transpose Convolution**

[1]Shelhamer et al, Fully Convolutional Networks for Semantic Segmentation, TPAMI 2016

Recall: Transpose Convolution

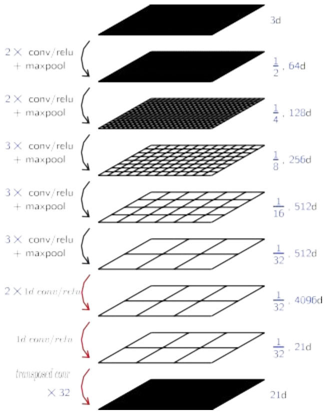
- Allows for learnable upsampling
- Also known as Deconvolution (bad) or Upconvolution
- Traditionally, we could achieve upsampling through interpolation or similar rules
- Why not allow the network to learn the rules by itself?
- Let us see a 1D example



Credit: Francois Fleuret

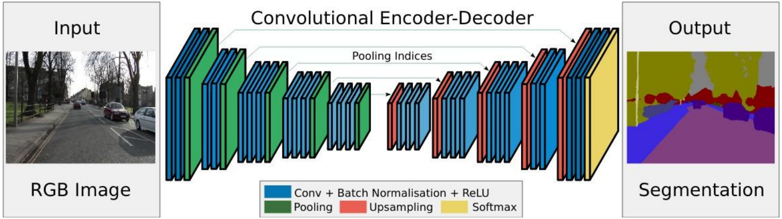
FCN with VGG-16 Backbone

- Remove fully connected layers, replace three FC layers with 1D conv layers;
- last layer has $C+1$ filters which give class probabilities
- Note that spatial size is downsampled because of maxpool operations
- One final upsampling layer to get back to original size



Credit: Francois Fleuret

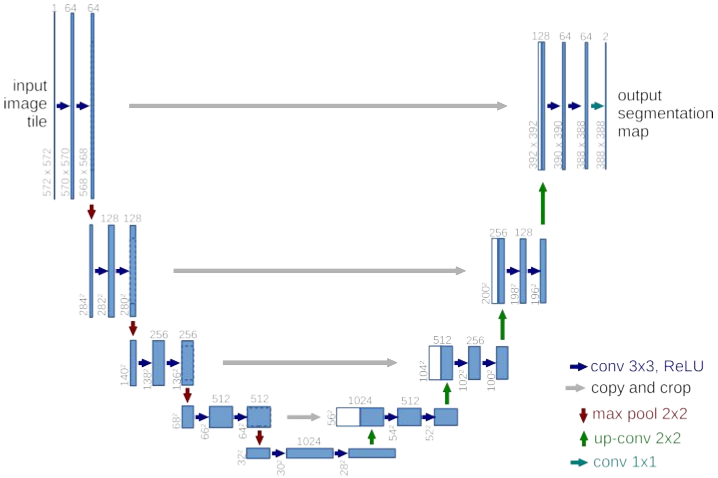
SegNet[2]



- Fully convolutional encoder-decoder architecture Encoder is VGG-16 without FC layers
- Decoder maps low-resolution encoder feature maps to input resolution.
- The final decoder output feature maps are fed to a softmax classifier for pixel-wise classification

U-Net Architecture

- One modification from FCN is: upsampling part has large number of feature channels
- Concatenation of feature maps from encoder gives localization information to decoder
- Designed for biomedical image segmentation;



FCN Challenges

What challenges could arise when using FCN on complex scenes?

- FCN may not learn that some visual patterns are co-occurent; E.g. cars are on roads, while boats are over rivers

- FCN may predict parts of an object as different categories; E.g. parts of a skyscraper may be mis-classified as different buildings

[4] Zhao et al, Pyramid Scene Parsing Network, CVPR 2017

PSPNet: Pyramid Scene Parsing Network[4]

What challenges could arise when using FCN on complex scenes?

- FCN may not learn that some visual patterns are co-occurent; E.g. cars are on roads, while boats are over rivers

- FCN may predict parts of an object as different categories; E.g. parts of a skyscraper may be mis-classified as different buildings

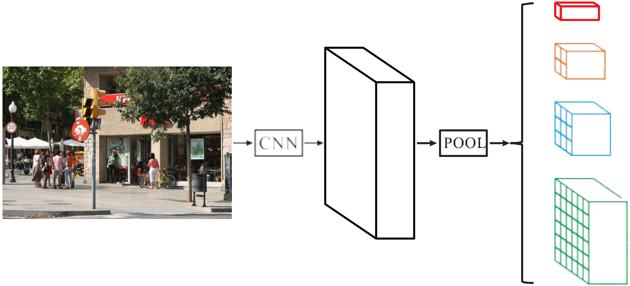
Hypothesis: Using **global context information** can lead to better segmentation; a network with suitable global scene-level information at different scales can improve segmentation

[4] Zhao et al, Pyramid Scene Parsing Network, CVPR 2017

PSPNet

- Incorporates pyramid pooling module to aggregate context at different scales.
- Global Context Integration:
    - Captures both global and local contextual information for precise segmentation.
- Flexibility in Backbone Architecture:
    - Compatible with various deep learning architectures like ResNet for feature extraction.

PSPNet: Pyramid Pooling Module



Average pooling done for four different scales

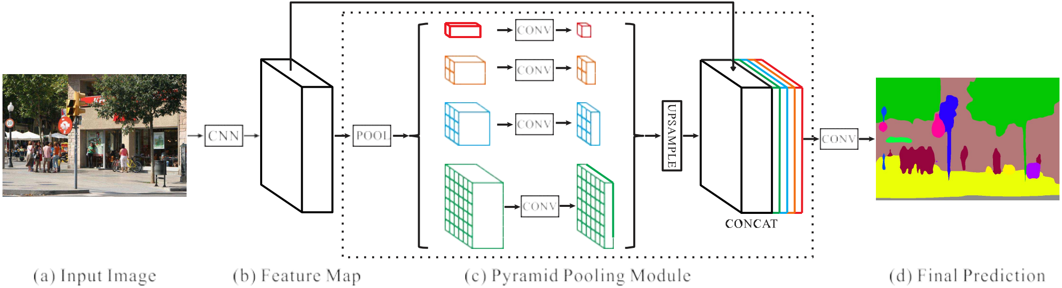
Take final layer feature map of a deep network like ResNet

Pyramid Pooling Module combines features in four different scales:

- Coarsest level is simply global average pooling
- Each successive pooling level gives increased localization information

Average pooled outputs are thus $1 \times 1 \times c$, $2 \times 2 \times c$, $3 \times 3 \times c$, $6 \times 6 \times c$ where c is number of input channels

Pyramid Pooling Module

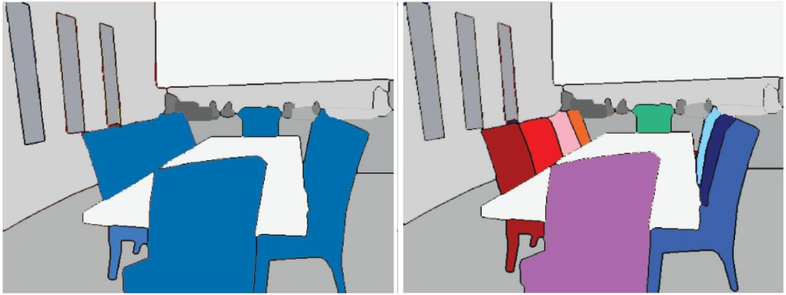


- Following average pooling, 1 × 1 conv layer reduce number of channels at each scale
- Low-dimension pooled maps are upsampled (can use bilinear interpolation) to same size as original
- Features then concatenated with original feature map
- Concatenated feature maps passed through upsampling and conv layers to generate segmentation

Performance and Applications of PSPNet

- Benchmark Achievements:
  - Demonstrates state-of-the-art performance on PASCAL VOC, and Cityscapes datasets.
- Versatility Across Scenes:
  - Effectively parses complex scenes with diverse object scales and relationships.
- Broad Applications:
  - Utilized in diverse fields such as autonomous driving, medical imaging, and urban planning.

Instance Segmentation



Semantic Segmentation

Semantic Instance Segmentation

Credit: [Soroush, StackOverFlow]

Mask R-CNN

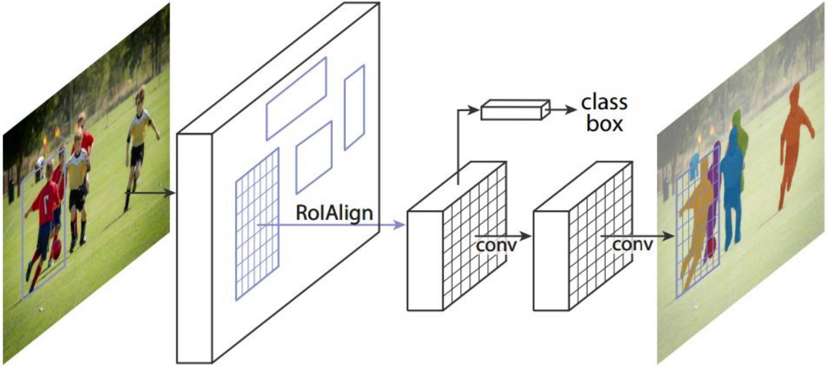
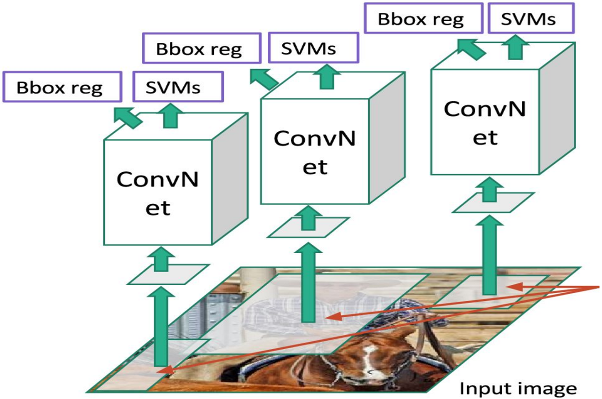


Figure 1. The **Mask R-CNN** framework for instance segmentation.

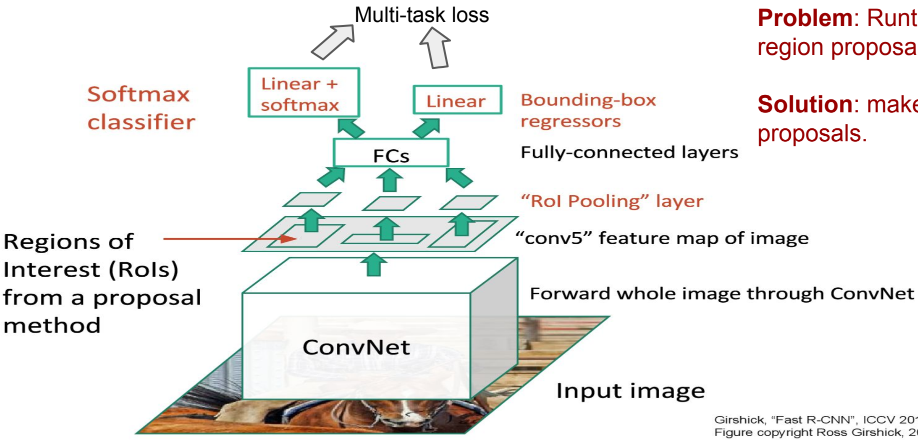
R-CNN



Problem: Training is **slow** (84h), takes a lot of disk space

***Solution**: Pass the image through convnet before cropping! Crop the conv feature instead!*

Fast R-CNN



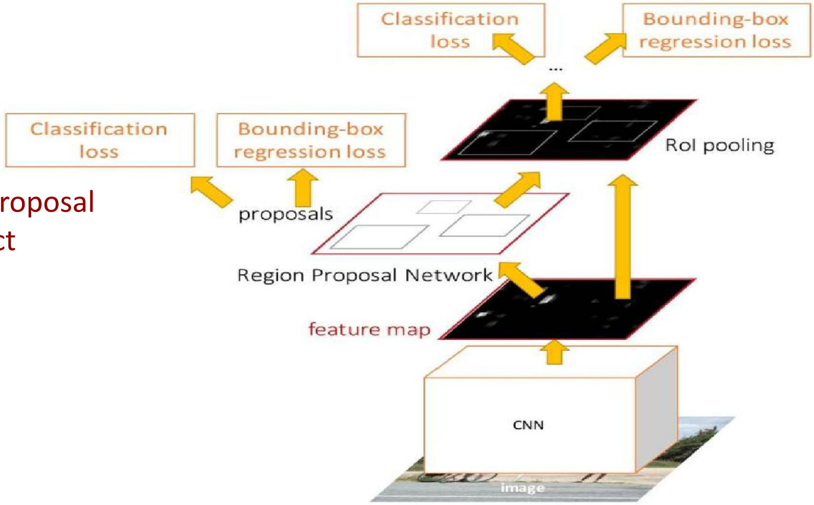
Problem: Runtime dominated by region proposals!

Solution: make CNN do region proposals.

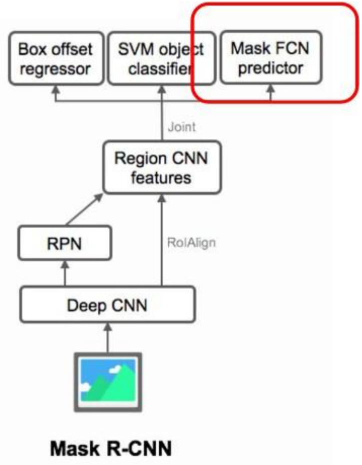
Girshick, "Fast R-CNN", ICCV 2015.

Faster R-CNN:

Two stages: region proposal generation and object classification



Mask R-CNN[7]



- Aims to tackle instance segmentation (where each pixel is given a class label, as well as an object ID)
- Extends Faster R-CNN by adding a branch for predicting object masks, enabling instance segmentation.
- Incorporates **RoIAlign** to accurately extract features for each RoI

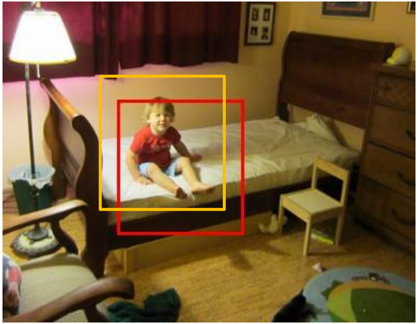
Credit: *[Lilian Weng, Github.io]*

[7] He, Mask R-CNN, ICCV 2017

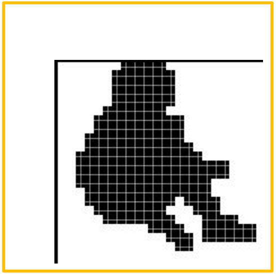
Mask R-CNN: RoIAlign

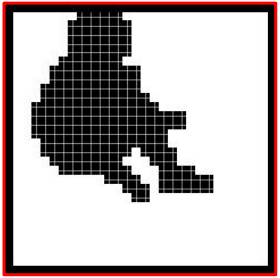
- An improvement of RoIPool operation (used in detection frameworks)
- There is loss of information when moving from object proposals in image space to proposals in feature space; Why does this matter?
- It is significant because one pixel in feature space is equivalent to many pixels on image
- RoIAlign performs bilinear interpolation for the exact coordinate. This preserves translation-equivariance of masks

Mask R-CNN: RoIAlign



target masks on RoIs

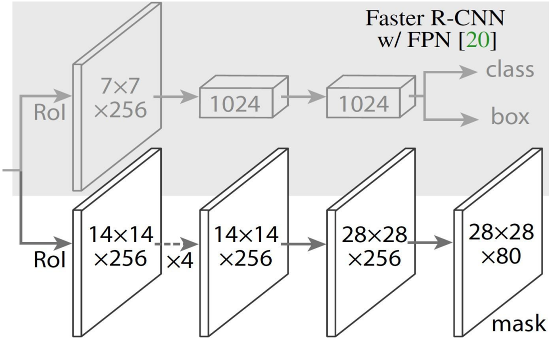




Translation of object in RoI => Same translation of mask in RoI

Credit: [Kaiming he, ICCV 2017 Tutorial]

Mask R-CNN: FCN Mask Head



- FCN branch generates mask for each proposal
- Mask is 28 × 28 in size during training
- Rescaled to bounding box size and overlaid on image during inference

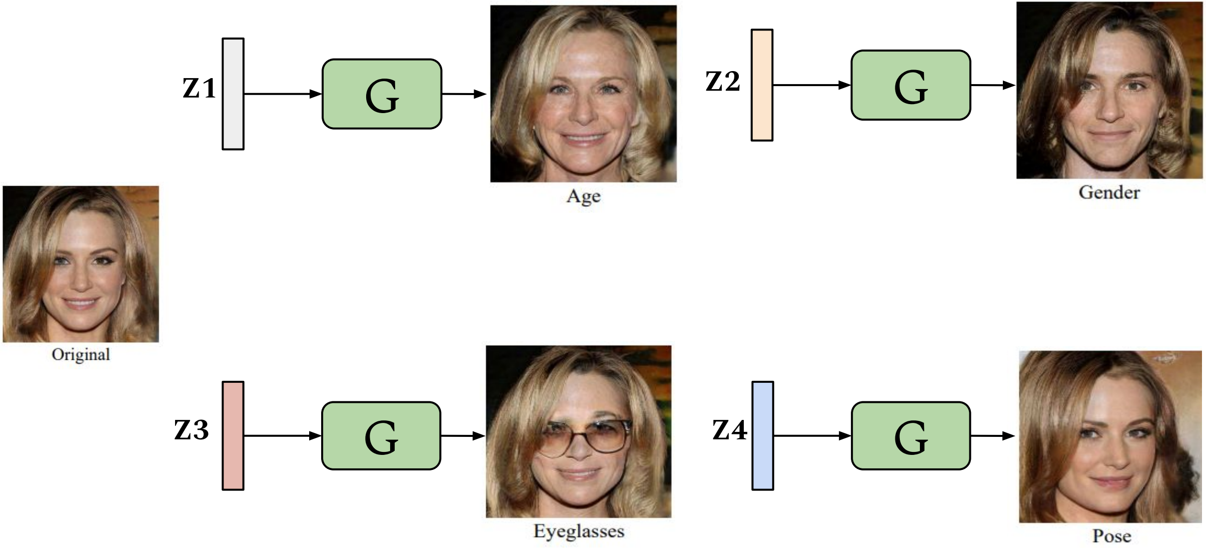
Credit: [Kaiming he, ICCV 2017 Tutorial]()

Mask R-CNN

- Instance Segmentation Precision: Provides high-precision segmentation masks for each detected object, distinguishing between instances of the same class.
- Benchmark Performance: Demonstrates state-of-the-art results on COCO dataset for both detection and segmentation tasks.

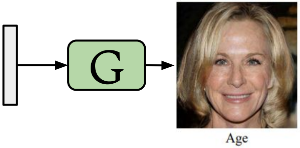
StyleGAN

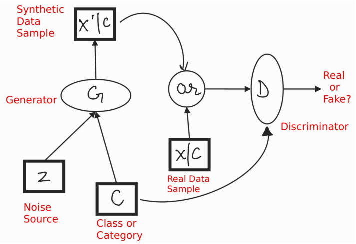
Controllable Generation



Controllable Generation vs Conditional Generation

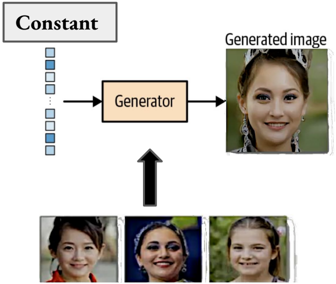
Controllable Generation	Conditional Generation
Examples with the features that you want	Examples from the classes you want
Training data does not need to be labeled	Training dataset needs to be labeled
Manipulate the input noise vector	Append a class vector to the input





StyleGAN

AIM



Style-based generator for

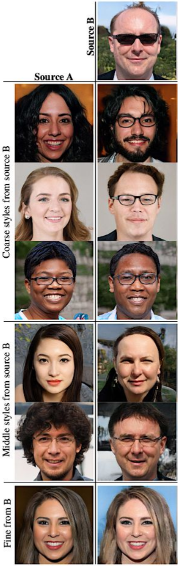
Style Mix and Matching

StyleGAN

High level features from Source B

Mid level features from Source B

Fine level features from Source B

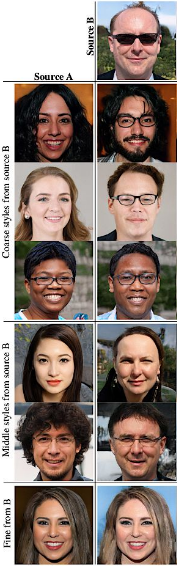


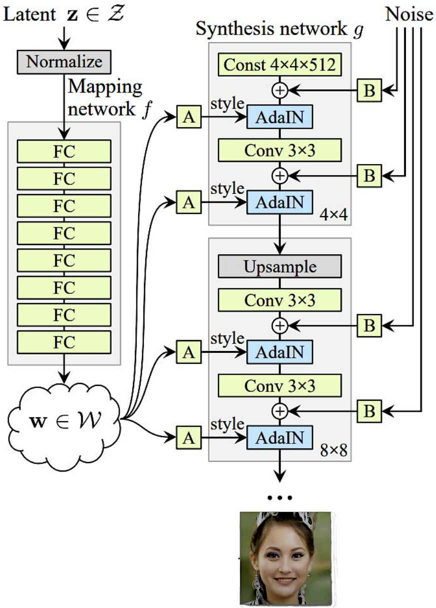
StyleGAN

High level features from Source B

Mid level features from Source B

Fine level features from Source B





ProGANs

ProGAN (Progressive GANs) is a new technique developed by NVIDIA Labs to improve both the speed and stability of GAN training.

<u>Key idea:</u> Progressive growing technique.

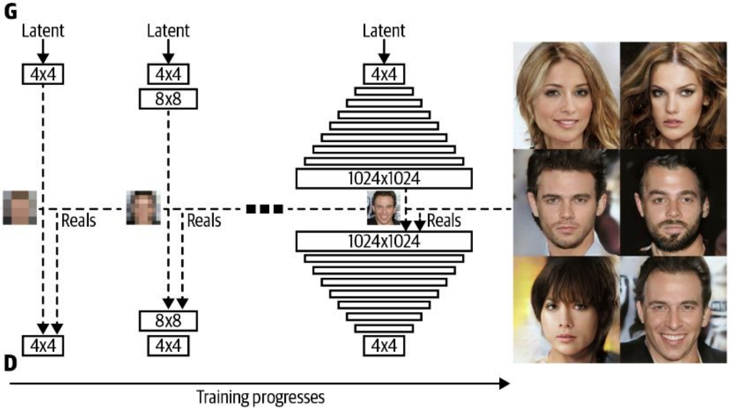
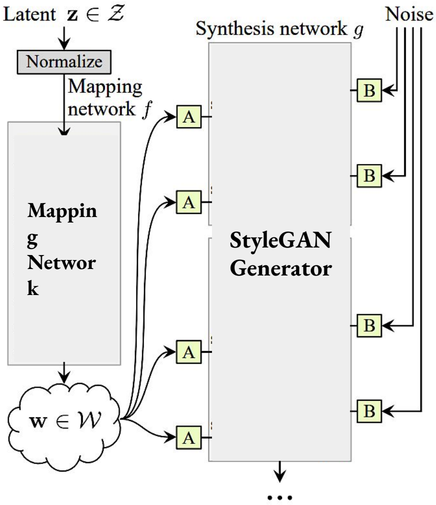


Figure 9-7. The Progressive GAN training mechanism, and some example generated faces[7]

Style GAN

Main Components of StyleGAN:

- Noise mapping network
- Progressive growing Generator
- Adaptive Instance Normalization (AdaIN)
- Style Variations

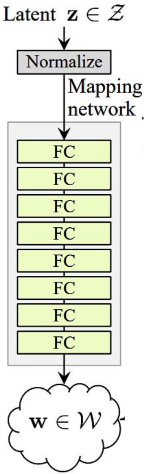


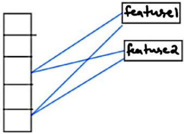
(b) Style-based generator

Style GAN

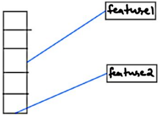
Main Components of StyleGAN:

- Noise mapping network





Z-Space
Entanglement

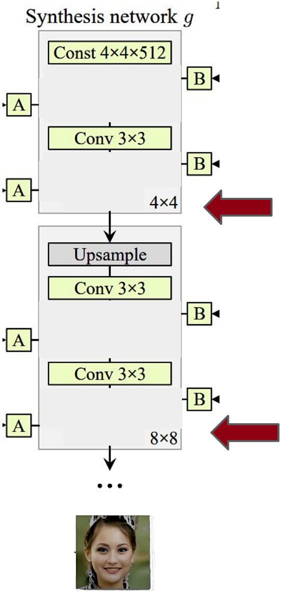


W-Space
Less Entanglement

Style GAN

Main Components of StyleGAN:

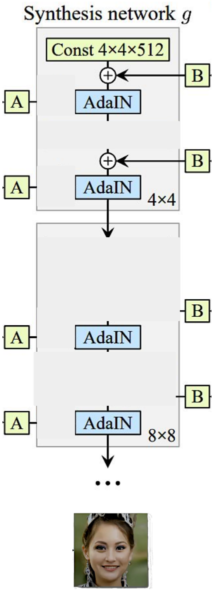
- Progressive growing Generator



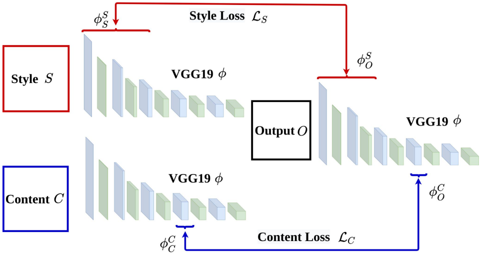
Style GAN

Main Components of StyleGAN:

- Adaptive Instance Normalization (AdaIN)



Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization



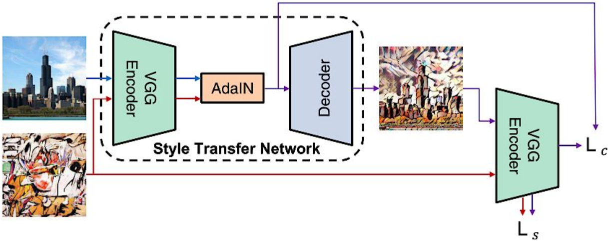
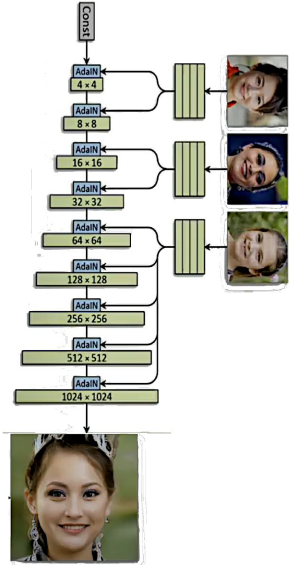


Figure 2. An overview of our style transfer algorithm. We use the first few layers of a fixed VGG-19 network to encode the content and style images. An AdaIN layer is used to perform style transfer in the feature space. A decoder is learned to invert the AdaIN output to the image spaces. We use the same VGG encoder to compute a content loss $\mathcal{L}_c$ (Equ. 12) and a style loss $\mathcal{L}_s$ (Equ. 13).

Huang, Xun, and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. Proceedings of the IEEE International Conference on Computer Vision. 2017.

Key Idea

(Latent code of one input with many styles to get diverse outputs)

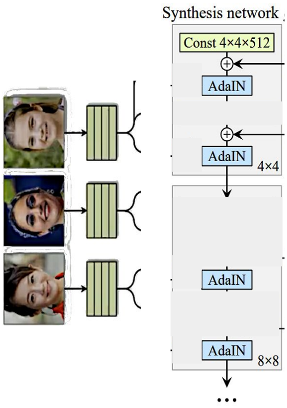


AdaIn at intermediate layers

https://www.youtube.com/watch?v=4IInDT_S0ow&t=4766s

Key Idea

(Latent code of one input with many styles to get diverse outputs)



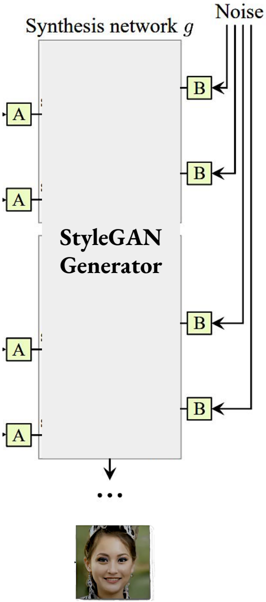
AdaIn at intermediate layers

https://www.youtube.com/watch?v=4IInDT_S0ow&t=4766s

Style GAN

Main Components of StyleGAN:

- Style Variations



Style Variations

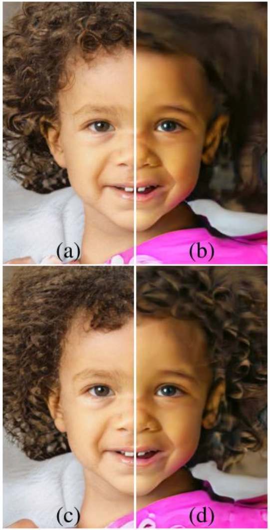


Figure 5. Effect of noise inputs at different layers of our generator. (a) Noise is applied to all layers. (b) No noise. (c) Noise in fine layers only (64^2 – 1024^2). (d) Noise in coarse layers only (4^2 – 32^2).

Style GAN

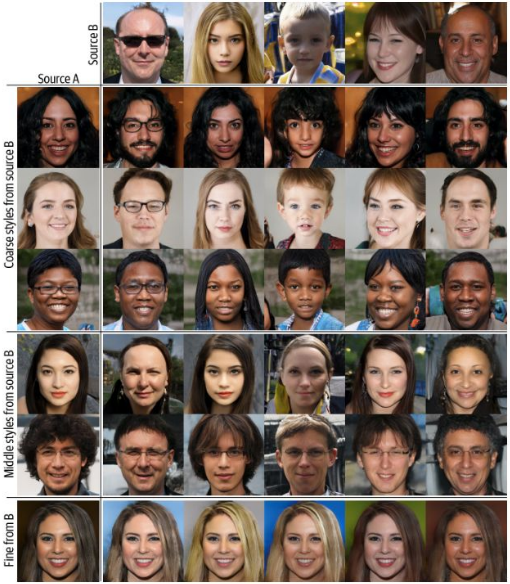


Figure 9-14. Merging styles between two generated images at different levels of detail[18]