Deep Learning

3. Computer Vision Tasks

Term 4, 2025

Skoltech

Outline

- A short recap of a previous lecture
- Classification evaluation metrics
- Edge detection task
- Image enhancement evaluation metrics
- Denoising and super-resolution task
- Segmentation evaluation metrics
- Semantic segmentation task
- Bounding box detection task
- Instance and panoptic segmentation tasks

A short recap of a previous lecture

- History of CNN development for an image classification task
- First layer extracts features
- Middle layers are the body of CNN
- The last linear layer(s) is the classification head
- Model input is an image
- Model output is a vector with NCLASS elements
- Each value in a vector is a confidence
- Top-1 value is the predicted class of the input image

Edge Detection Task

- Mark every pixel as an edge or not an edge
- Binary classification problem for every pixel
- Need to learn quality metrics for classification at first
  - Top-1, Top-K accuracy
  - Precision
  - Recall
  - F1 score
  - AP (Average Precision)
  - mAP (mean Average Precision)

Classification Evaluation Metrics

Basic terms:

- There are several objects
  - We want to categorize them (e.g., define a class for every object)
  - Some problem setups allow multiclass labeling
- For every object there is a vector of values
  - Each value is a confidence, that a given object belongs to a given class
  - There might be several such vectors (due to multiclass labeling)
- We may select not a single, but several largest confidence values per vector
  - In this case a guess is correct, if true class is one of the chosen classes

Classification Evaluation Metrics

- Top-1 Accuracy: percentage of objects where the top predicted class matches the true class
  - If a model predicts:
    - Image 1: "cat" (true: "cat") → Correct
    - Image 2: "dog" (true: "cat") → Incorrect
    - Image 3: "bird" (true: "bird") → Correct
  - Then:
    - Top-1 Accuracy = 2/3 = 66.7%
- Top-K Accuracy: percentage of objects where the top K predicted classes contain the true class
  - It is just a single value

Classification Evaluation Metrics

- For every class, the following metrics can be computed:
  - True Positive (TP): Correctly identified instances (e.g., correctly detected cats)
  - False Positive (FP): Incorrectly identified instances (e.g., non-cats detected as cats)
  - False Negative (FN): Missed instances (e.g., actual cats that were not detected)
  - True Negative (TN): Correctly rejected instances (e.g., correctly identified non-cats)

Confusion Matrix	Predicted +	Predicted -
Actual +	TP	FN
Actual -	FP	TN

Classification Evaluation Metrics

- Precision (P): the ratio of correctly identified positive instances to all instances identified as positive
  - P = TP / (TP + FP)
  - P@K means our guess contains K classes for every object
  - Minimum is 0
  - Maximum is 1
  - The more, the better
  - A vector with number of classes values

Confusion Matrix	Predicted +	Predicted -
Actual +	TP	FN
Actual -	FP	TN

Classification Evaluation Metrics

- Precision (P): the ratio of correctly identified positive instances to all instances identified as positive
  - P = TP / (TP + FP)
  - If a model detects 100 edges and 80 of them are actual edges, then:
    - TP = 80 (correctly detected edges)
    - FP = 20 (non-edges detected as edges)
    - P = 80 / (80 + 20) = 0.8 or 80%

Confusion Matrix	Predicted +	Predicted -
Actual +	TP	FN
Actual -	FP	TN

Classification Evaluation Metrics

- Recall (R): the ratio of correctly identified positive instances to all actual positive instances
  - R = TP / (TP + FN)
  - R@K means our guess contains K classes for every object
  - Minimum is 0
  - Maximum is 1
  - The more, the better
  - A vector with number of classes values

Confusion Matrix	Predicted +	Predicted -
Actual +	TP	FN
Actual -	FP	TN

Classification Evaluation Metrics

- Recall (R): the ratio of correctly identified positive instances to all actual positive instances
  - R = TP / (TP + FN)
  - If there are 100 actual edges in an image and the model detects 70 of them:
    - TP = 70 (correctly detected edges)
    - FN = 30 (missed edges)
    - R = 70 / (70 + 30) = 0.7 or 70%

Confusion Matrix	Predicted +	Predicted -
Actual +	TP	FN
Actual -	FP	TN

Classification Evaluation Metrics

- F1-score: harmonic mean of precision and recall, providing a balanced measure of a model's performance
  - F1 = 2 * (precision * recall) / (precision + recall)
  - F1@K means our guess contains K classes for every object
  - Minimum is 0
  - Maximum is 1
  - The more, the better
  - A vector with number of classes values

Confusion Matrix	Predicted +	Predicted -
Actual +	TP	FN
Actual -	FP	TN

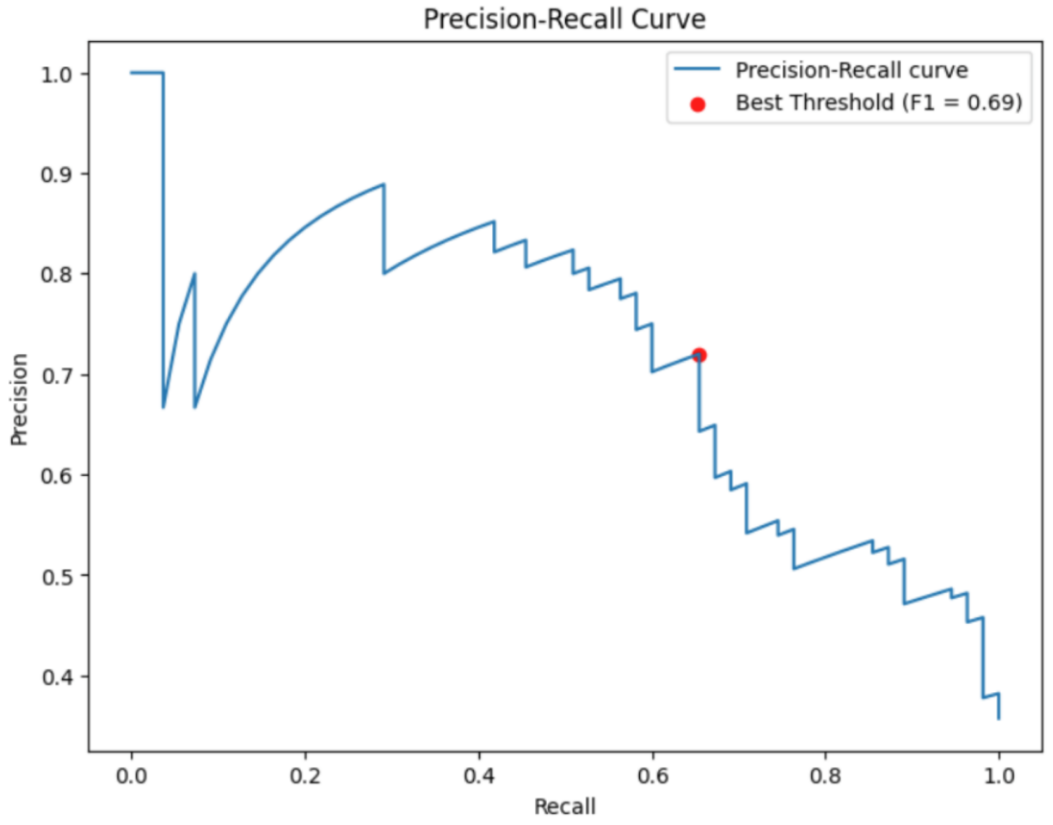
Classification Evaluation Metrics

- F1-score: harmonic mean of precision and recall, providing a balanced measure of a model's performance
  - F1 = 2 * (precision * recall) / (precision + recall)
  - Using the previous examples:
    - precision = 0.8
    - recall = 0.7
    - F1 = 2 * (0.8 * 0.7) / (0.8 + 0.7) = 0.74666

Confusion Matrix	Predicted +	Predicted -
Actual +	TP	FN
Actual -	FP	TN

Classification Evaluation Metrics

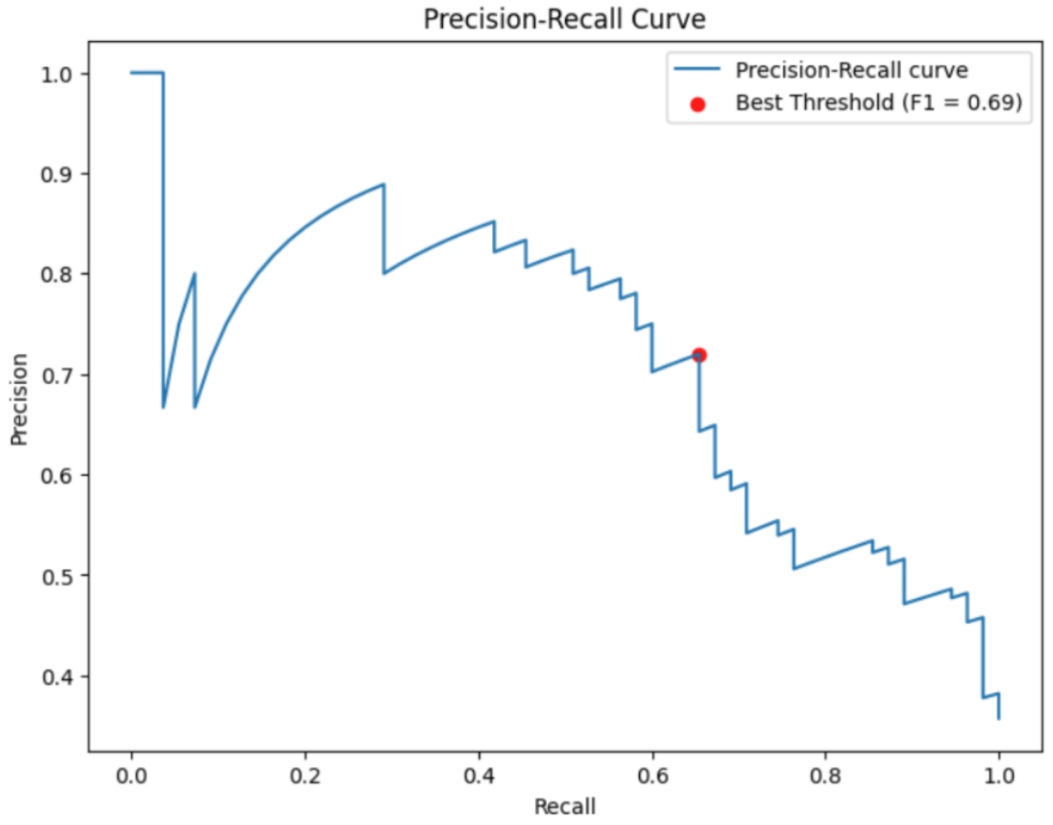
- Recall levels: different confidence thresholds for classification
  - As threshold decreases:
    - More detections (higher recall)
    - Generally, more false positives (lower precision)
  - As threshold increases:
    - Fewer detections (lower recall)
    - Generally, more accurate detections (higher precision)
  - Confidence threshold acts similar to labeling in a top-K style



https://www.blog.trainindata.com/precision-recall-curves/

Classification Evaluation Metrics

- AP: average precision
  - It is not an average of a precision over all classes!
  - It is an area under a Precision-Recall curve (average over all possible recalls)
  - How to calculate:
    1. Sort detections by confidence score
    2. Calculate precision and recall at each confidence threshold
    3. Plot precision-recall curve
    4. Calculate area under the curve (AUC)



https://www.blog.trainindata.com/precision-recall-curves/

Classification Evaluation Metrics

- AP: average precision
  - It is not an average of a precision over all classes!
  - It is an area under a Precision-Recall curve (average over all possible recalls)
  - How to calculate:
    1. **Sort detections by confidence score**
    2. Calculate precision and recall at each confidence threshold
    3. Plot precision-recall curve
    4. Calculate area under the curve (AUC)

Detection	Confidence	TP/FP
Edge	0.95	TP
Edge	0.90	TP
Edge	0.85	FP
Edge	0.80	TP
Edge	0.75	FP
Edge	0.70	TP

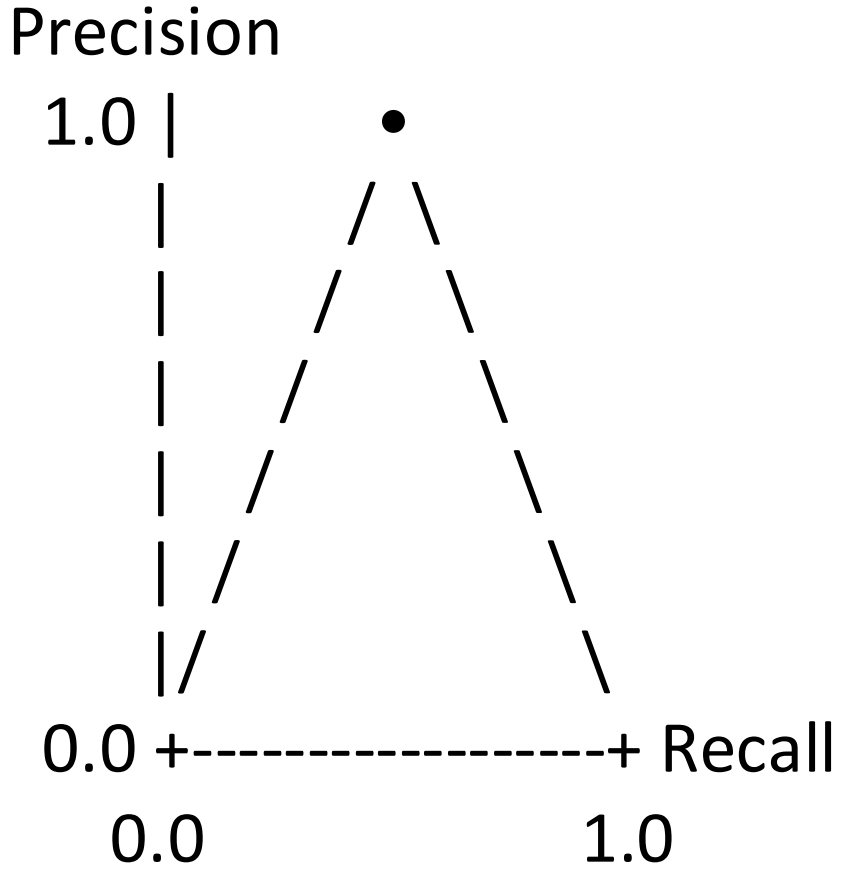
Classification Evaluation Metrics

- AP: average precision
  - It is not an average of a precision over all classes!
  - It is an area under a Precision-Recall curve (average over all possible recalls)
  - How to calculate:
    1. Sort detections by confidence score
    2. **Calculate precision and recall at each confidence threshold**
    3. Plot precision-recall curve
    4. Calculate area under the curve (AUC)

Threshold	TP	FP	Precision	Recall
0.95	1	0	1.0	0.1
0.90	2	0	1.0	0.2
0.85	2	1	0.67	0.2
0.80	3	1	0.75	0.3
0.75	3	2	0.60	0.3
0.70	4	2	0.67	0.4

Classification Evaluation Metrics

- AP: average precision
  - It is not an average of a precision over all classes!
  - It is an area under a Precision-Recall curve (average over all possible recalls)
  - How to calculate:
    1. Sort detections by confidence score
    2. Calculate precision and recall at each confidence threshold
    3. **Plot precision-recall curve**
    4. **Calculate area under the curve (AUC)**



Classification Evaluation Metrics

- mAP (Mean Average Precision): average precision across all classes
  - For object detection with 2 classes:
    - Class "Cat":
      - AP = 0.75
    - Class "Dog":
      - AP = 0.85
    - mAP = (0.75 + 0.85) / 2 = 0.80

Edge Detection Task

- Edge detection: identifying points in an image where brightness changes sharply or has discontinuities
- Edge detection: per-pixel classification problem
- CNN approaches:
  - HED (Holistically-Nested Edge Detection) (2015)
  - RCF (Richer Convolutional Features) (2017)

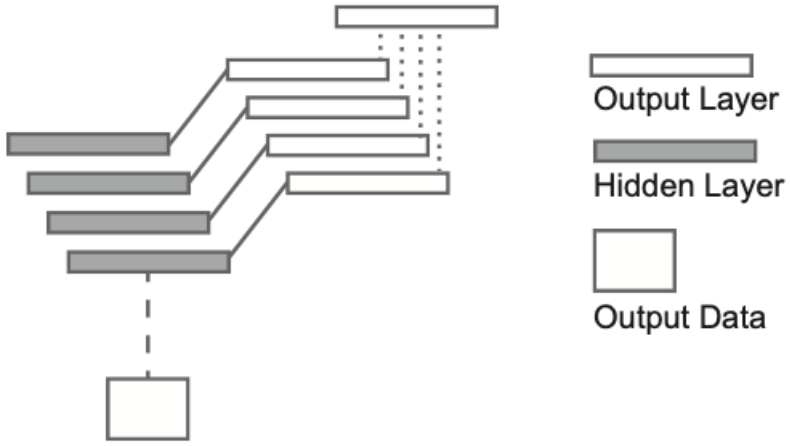
Method	F1-Score on BSD500 Dataset
Canny	0.61
HED	0.79
RCF	0.81

Edge Detection Task

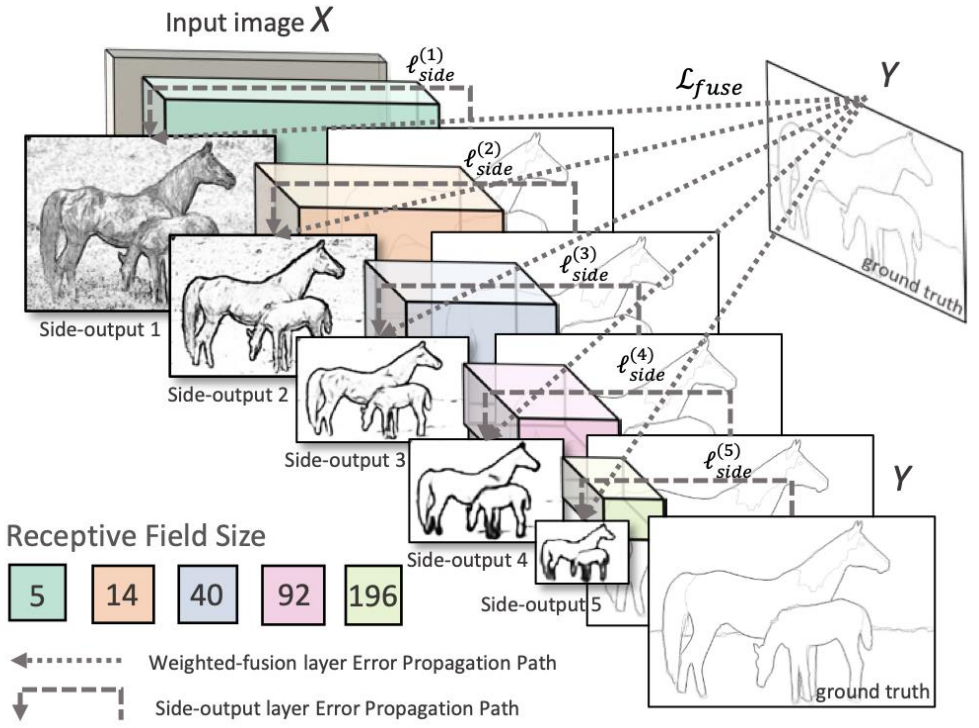
- HED (Holistically-Nested Edge Detection) (2015)
  - Key Innovations:
    - First to use deep learning for edge detection
    - Combines multi-scale and multi-level features through side outputs
    - Uses a weighted fusion layer to combine predictions from different scales
  - Architecture:
    - VGG-16 backbone with side outputs after each stage
    - Each side output produces an edge map
    - Final fusion layer combines all side outputs
  - Training: Uses a weighted cross-entropy loss for each side output
  - https://arxiv.org/abs/1504.06375

Edge Detection Task

- HED (Holistically-Nested Edge Detection) (2015)



https://arxiv.org/abs/1504.06375



https://arxiv.org/abs/1504.06375

Edge Detection Task

- RCF (Richer Convolutional Features) (2017)
  - Key Innovations:
    - Uses all convolutional features in the network, not just the last layer in a stage
    - Introduces a fusion strategy that preserves more detailed information
    - Better handles multi-scale edge detection
    - More robust to different edge types and scales
  - Architecture:
    - VGG-16 backbone with feature fusion at multiple levels
    - Combines features from all convolutional layers
    - Uses a hierarchical feature fusion strategy
  - Training: Uses a balanced loss function to handle edge/non-edge class imbalance
  - https://arxiv.org/abs/1612.02103

Edge Detection Task

- RCF (Richer Convolutional Features) (2017)

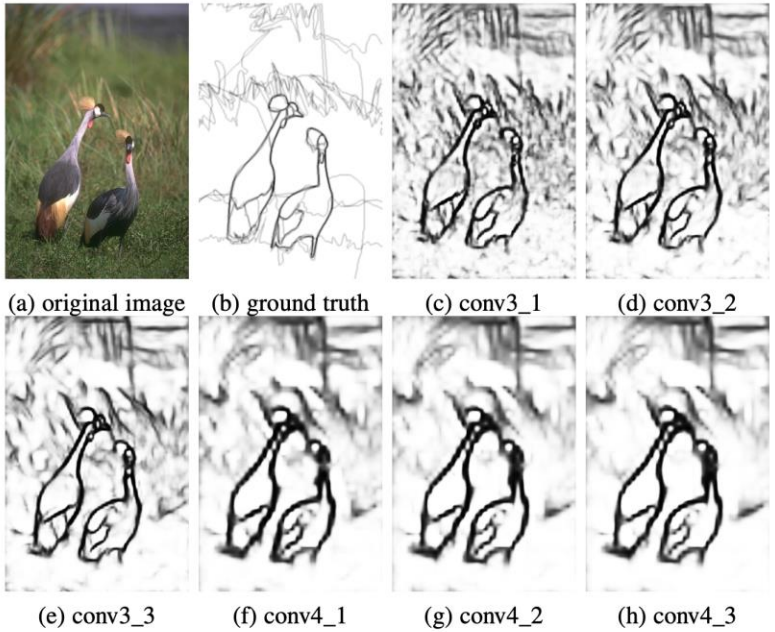


Fig. 1: We build a simple network based on VGG16 [11] to produce side outputs (c-h). One can see that convolutional features become coarser gradually, and the intermediate layers (c,d,f,g) contain essential fine details that do not appear in other layers.

https://arxiv.org/abs/1612.02103

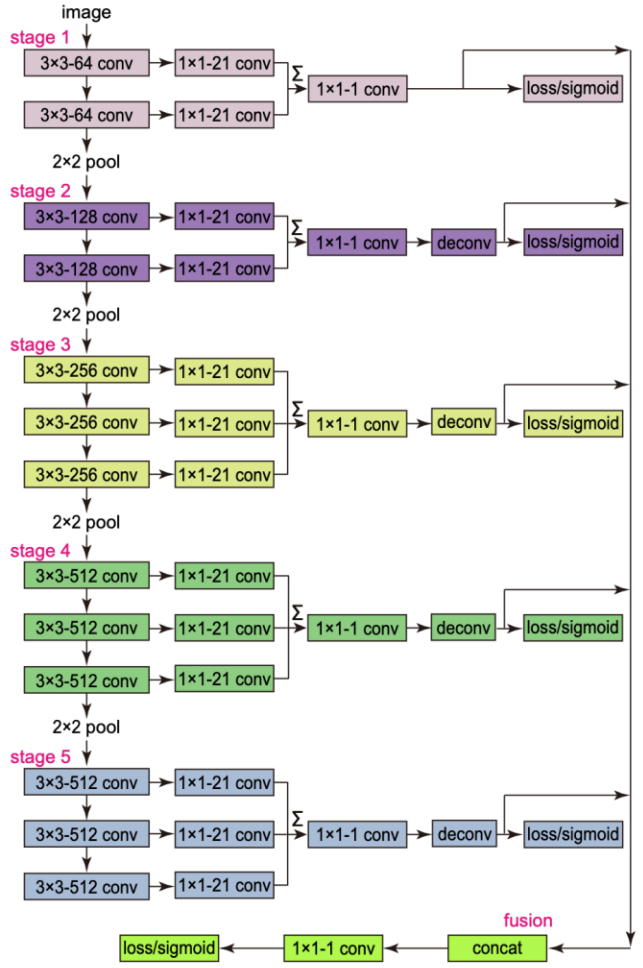


Fig. 2: Our RCF network architecture. The input is an image with arbitrary sizes, and our network outputs an edge possibility map in the same size.

https://arxiv.org/abs/1612.02103

Edge Detection Task

- Datasets
  - BSDS500 (Berkeley Segmentation Data Set and Benchmarks)
    - 500 natural images
    - Multiple human-annotated ground truth segmentations
    - Contains both edge and region annotations
    - https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html
  - NYUDv2 (NYU Depth Dataset V2)
    - 1449 RGB-D images
    - Includes depth information
    - Useful for studying edge detection with depth cues
    - https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html

Edge Detection Task

- Data augmentation
  - Geometric Transformations
    - Rotation, Scaling, Flipping (horizontal/vertical)
    - Perspective transforms
  - Photometric Transformations
    - Brightness, contrast adjustment
    - Color jittering
    - Gaussian noise addition
  - Synthetic Edge Generation
    - Add synthetic edges with known ground truth
    - Vary edge thickness and intensity
    - Create edge patterns

Denoising Task

- Find out per-pixel noise and subtract it from the original image
- Need to learn quality metrics for image denoising at first
  - PSNR
  - SSIM

Image Enhancement Evaluation Metrics

- PSNR (Peak Signal-to-Noise Ratio): ratio between the maximum possible power of a signal and the power of corrupting noise
  - PSNR = 20 * log10(MAX_I) - 10 * log10(MSE)
    - MAX_I = maximum possible pixel value (usually 255)
    - MSE = mean squared error between original and processed image
  - For an 8-bit image (MAX_I = 255):
    - If MSE = 100
    - PSNR = 20 * log10(255) - 10 * log10(100) ≈ 48.13 - 20 = 28.13 dB
  - Interpretation:
    - Higher PSNR = better quality
    - PSNR > 30 dB: Good quality
    - PSNR < 20 dB: Poor quality

Image Enhancement Evaluation Metrics

- SSIM (Structural Similarity Index): Measures the similarity between two images based on luminance, contrast, and structure
  - $\mathrm{SSIM}(x, y) = [l(x, y)]^{\alpha} \cdot [c(x, y)]^{\beta} \cdot [s(x, y)]^{\gamma}$
    - $l(x, y)$ is a luminance comparison between samples x and y
    - $c(x, y)$ is a contrast comparison between samples x and y
    - $s(x, y)$ is a structure comparison between samples x and y
    - Each measure is just a combination of mean, variance and covariance values
  - Interpretation:
    - SSIM = 1: Perfect match
    - SSIM = 0: No structural similarity
    - Typical range: 0.8-0.95 for good quality

Denoising Task

- Image denoising: removing noise from an image while preserving important features
- CNN approaches:
  - DnCNN (Denoising Convolutional Neural Networks) (2017)
  - FFDNet (Fast and Flexible denoising network) (2018)

Denoising Task

- DnCNN (Denoising Convolutional Neural Networks) (2017)
  - Key Innovations:
    - Introduces residual learning to predict noise instead of clean image
    - Handles different noise levels with a single model
    - Uses batch normalization and ReLU for better training
  - Architecture:
    - 17-layer CNN with residual connections
    - Uses batch normalization after each convolution
    - Outputs noise map which is subtracted from input
  - Training: uses mean squared error loss on noise maps
  - https://arxiv.org/abs/1608.03981

Denoising Task

- DnCNN (Denoising Convolutional Neural Networks) (2017)

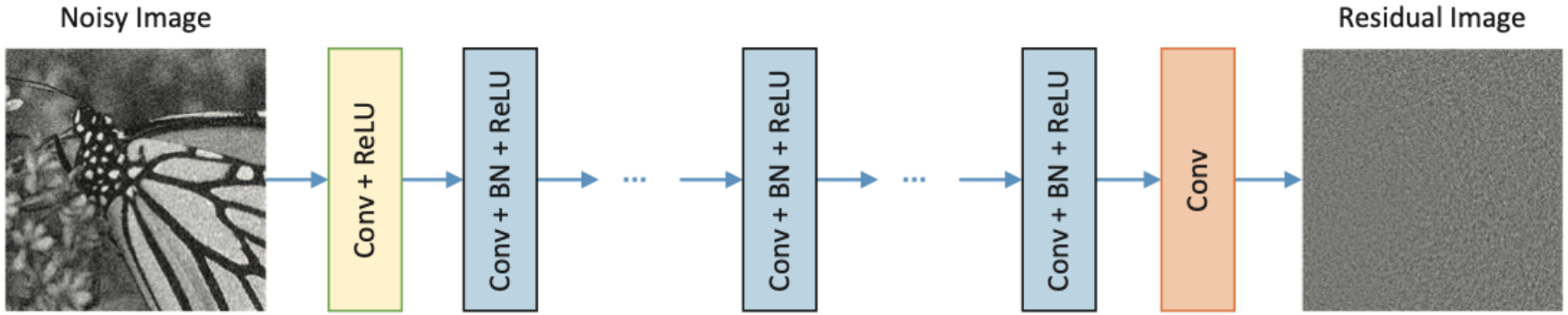


Fig. 1. The architecture of the proposed DnCNN network.

https://arxiv.org/abs/1608.03981

Denoising Task

- DnCNN (Denoising Convolutional Neural Networks) (2017)

Single Image Super-Resolution				
Dataset	Upscaling Factor	TNRD	VDSR	DnCNN-3
		PSNR / SSIM	PSNR / SSIM	PSNR / SSIM
Set5	2	36.86 / 0.9556	37.56 / **0.9591**	**37.58** / 0.9590
	3	33.18 / 0.9152	33.67 / 0.9220	**33.75** / **0.9222**
	4	30.85 / 0.8732	31.35 / **0.8845**	**31.40** / **0.8845**
Set14	2	32.51 / 0.9069	33.02 / **0.9128**	**33.03** / **0.9128**
	3	29.43 / 0.8232	29.77 / 0.8318	**29.81** / **0.8321**
	4	27.66 / 0.7563	27.99 / 0.7659	**28.04** / **0.7672**
BSD100	2	31.40 / 0.8878	31.89 / **0.8961**	**31.90** / **0.8961**
	3	28.50 / 0.7881	28.82 / 0.7980	**28.85** / **0.7981**
	4	27.00 / 0.7140	27.28 / **0.7256**	**27.29** / 0.7253
Urban100	2	29.70 / 0.8994	**30.76** / **0.9143**	30.74 / 0.9139
	3	26.42 / 0.8076	27.13 / **0.8283**	**27.15** / 0.8276
	4	24.61 / 0.7291	25.17 / **0.7528**	**25.20** / 0.7521

https://arxiv.org/abs/1608.03981

Denoising Task

- FFDNet (Fast and Flexible denoising network) (2018)
  - Key Innovations:
    - First to use noise level map as input
    - Can handle spatially variant noise
    - More flexible than previous methods
    - Faster inference time
  - Architecture:
    - Takes both noisy image and noise level map as input
    - Uses downsampling and upsampling for efficiency
    - Processes noise level map through separate branches
  - Training: uses noise level map to guide denoising process
  - https://arxiv.org/abs/1710.04026

Denoising Task

- FFDNet (Fast and Flexible denoising network) (2018)

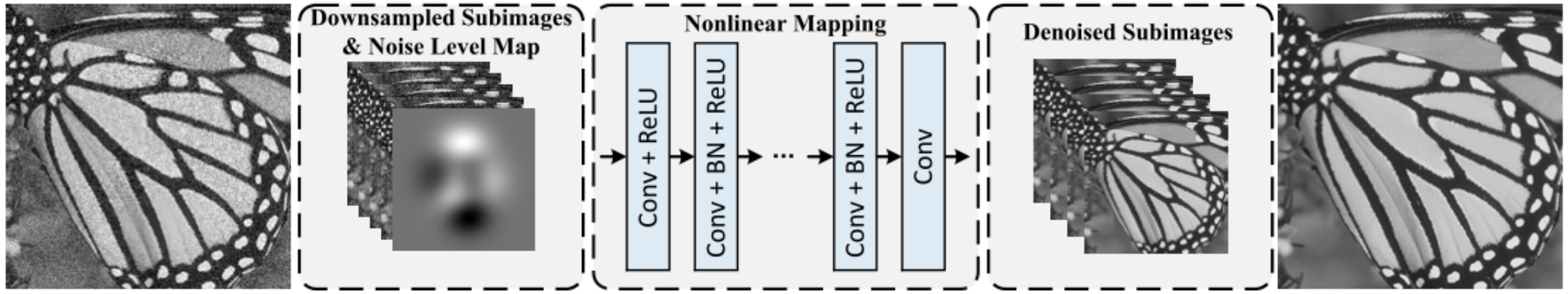


Fig. 1. The architecture of the proposed FFDNet for image denoising. The input image is reshaped to four sub-images, which are then input to the CNN together with a noise level map. The final output is reconstructed by the four denoised sub-images.

https://arxiv.org/abs/1710.04026

Denoising Task

- FFDNet (Fast and Flexible denoising network) (2018)
  - PSNR comparison against other approaches:

Images	*C.man*	*House*	*Peppers*	*Starfish*	*Monarch*	*Airplane*	*Parrot*	*Lena*	*Barbara*	*Boat*	*Man*	*Couple*	*Average*
Noise Level	$\sigma = 15$												
BM3D	31.91	34.93	32.69	31.14	31.85	31.07	31.37	34.26	33.10	32.13	31.92	32.10	32.37
WNNM	32.17	35.13	32.99	31.82	32.71	31.39	31.62	34.27	33.60	32.27	32.11	32.17	32.70
TNRD	32.19	34.53	33.04	31.75	32.56	31.46	31.63	34.24	32.13	32.14	32.23	32.11	32.50
DnCNN	32.61	34.97	33.30	32.20	33.09	31.70	31.83	34.62	32.64	32.42	32.46	32.47	32.86
FFDNet	32.42	35.01	33.10	32.02	32.77	31.58	31.77	34.63	32.50	32.35	32.40	32.45	32.75
Noise Level	$\sigma = 25$												
BM3D	29.45	32.85	30.16	28.56	29.25	28.42	28.93	32.07	30.71	29.90	29.61	29.71	29.97
WNNM	29.64	33.22	30.42	29.03	29.84	28.69	29.15	32.24	31.24	30.03	29.76	29.82	30.26
MLP	29.61	32.56	30.30	28.82	29.61	28.82	29.25	32.25	29.54	29.97	29.88	29.73	30.03
TNRD	29.72	32.53	30.57	29.02	29.85	28.88	29.18	32.00	29.41	29.91	29.87	29.71	30.06
DnCNN	30.18	33.06	30.87	29.41	30.28	29.13	29.43	32.44	30.00	30.21	30.10	30.12	30.43
FFDNet	30.06	33.27	30.79	29.33	30.14	29.05	29.43	32.59	29.98	30.23	30.10	30.18	30.43

https://arxiv.org/abs/1710.04026

Denoising Task

- Datasets
  - DIV2K
    - 1000 high-quality RGB images
    - Used for both denoising and super-resolution
    - Includes various noise levels
    - https://data.vision.ee.ethz.ch/cvl/DIV2K/
  - SIDD (Smartphone Image Denoising Dataset)
    - Real noisy images from smartphones
    - Paired clean images
    - Different lighting conditions and scenes
    - https://www.eecs.yorku.ca/~kamel/sidd/

Denoising Task

- Data augmentation
  - Noise Generation
    - Different noise types, including mixes
  - Noise Level Variation
    - Different noise levels for same image
    - Progressive noise addition
    - Noise pattern variation
  - Image Processing
    - Blur addition
    - JPEG compression artifacts
    - Sensor noise simulation

Super-Resolution Task

- Image super-resolution: increasing the resolution of an image while preserving or enhancing its quality
- CNN approach:
  - SRCNN (Super-Resolution CNN) (2014)
- Other neural networks:
  - ESRGAN (Enhanced Super-Resolution GAN) (2018)
  - RCAN (Residual Channel Attention Network) (2018)

Super-Resolution Task

- SRCNN (Super-Resolution CNN) (2014)
  - Key Innovations:
    - The first CNN for super-resolution
    - End-to-end mapping from low to high resolution
    - Three-layer architecture
    - Learns non-linear mapping
  - Architecture:
    - Patch extraction and representation
    - Non-linear mapping
    - Reconstruction
  - Training: uses MSE loss
  - https://arxiv.org/abs/1501.00092

Super-Resolution Task

- SRCNN (Super-Resolution CNN) (2014)

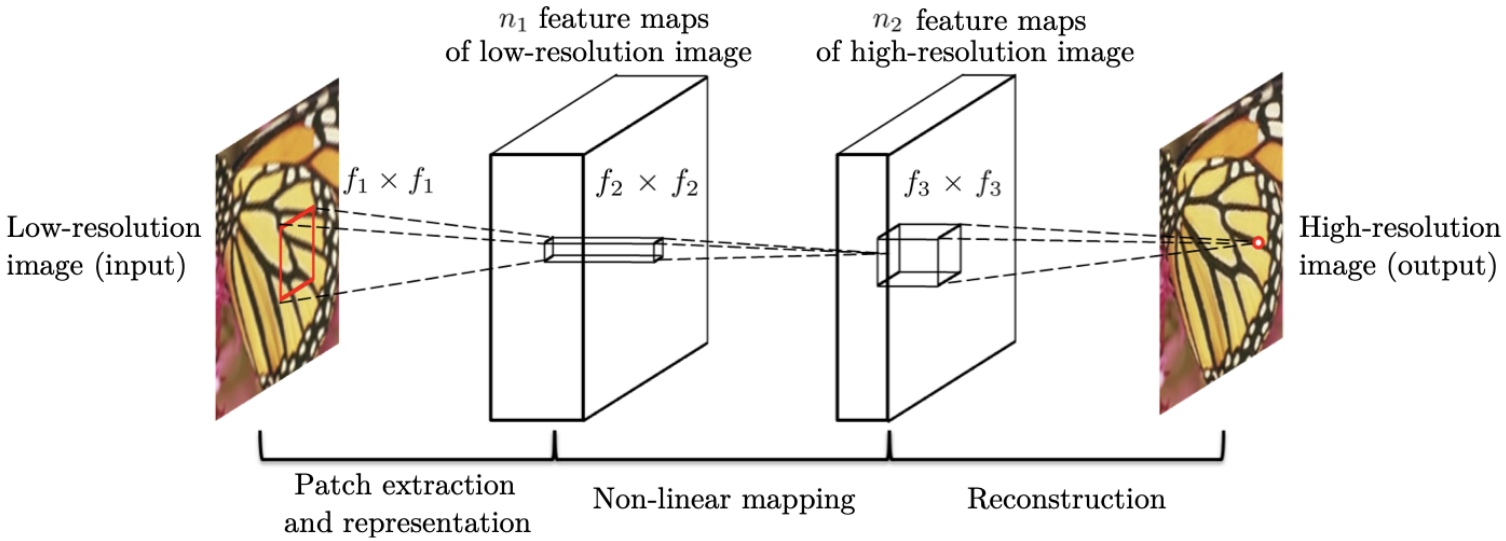


Fig. 2. Given a low-resolution image $\mathbf{Y}$, the first convolutional layer of the SRCNN extracts a set of feature maps. The second layer maps these feature maps nonlinearly to high-resolution patch representations. The last layer combines the predictions within a spatial neighbourhood to produce the final high-resolution image $F(\mathbf{Y})$.

https://arxiv.org/abs/1501.00092

Super-Resolution Task

- SRCNN (Super-Resolution CNN) (2014)

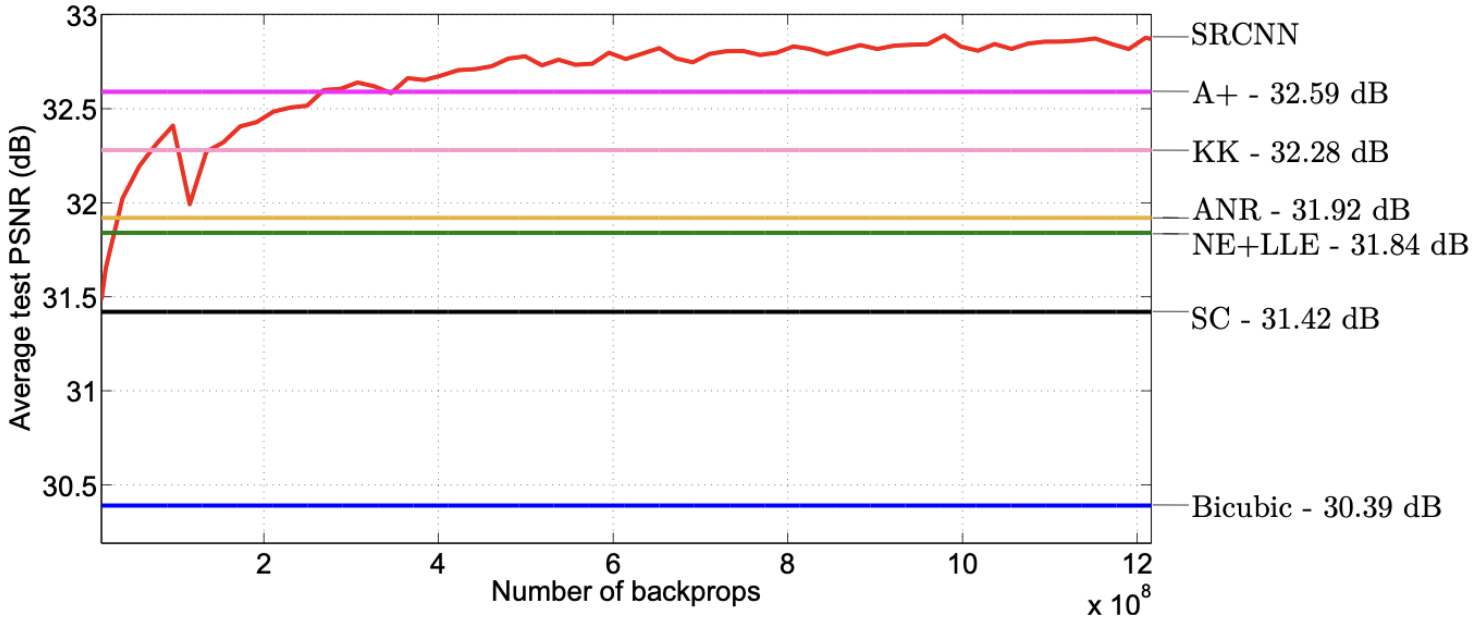


Fig. 10. The test convergence curve of SRCNN and results of other methods on the Set5 dataset.

https://arxiv.org/abs/1501.00092

Super-Resolution Task

- Datasets:
  - DIV2K
    - 1000 high-quality images, various scales (2x, 3x, 4x), used for training and validation
    - https://data.vision.ee.ethz.ch/cvl/DIV2K/
  - Set5/Set14
    - Small test sets (5/14 images), Commonly used for benchmarking
    - https://huggingface.co/datasets/eugenesiow/Set5
    - https://huggingface.co/datasets/eugenesiow/Set14
  - Urban100
    - 100 urban scene images, challenging for super-resolution, contains complex structures
    - https://huggingface.co/datasets/eugenesiow/Urban100

Super-Resolution Task

- Data Augmentation for Super-Resolution:
  - Data augmentation approaches for image denoising
    - Noise addition
    - JPEG artifacts
    - Geometric transformations
  - Downsampling Methods
    - Bicubic downsampling
    - Gaussian blur + downsampling
    - Different downsampling kernels

Super-Resolution Task

- Applications:
  - Medical Imaging
    - Enhancing medical scans
    - Improving diagnosis accuracy
    - Reducing scanning time
  - Satellite Imagery
    - Enhancing remote sensing data
    - Improving land use analysis
    - Better environmental monitoring
  - Video Enhancement
    - Upscaling old videos
    - Improving streaming quality
    - Enhancing surveillance footage
  - Mobile Photography
    - Zoom enhancement
    - Low-light image improvement
    - Better detail preservation

Segmentation Task

- Partitioning an image into multiple segments or regions
- Need to learn quality metrics for segmentation at first
  - IoU (Intersection over Union)
  - mIoU (mean Intersection over Union)

Segmentation Evaluation metrics

- IoU (Intersection over Union): Ratio of the area of overlap to the area of union between predicted and ground truth segments
  - IoU = Area of Overlap / Area of Union
  - Ground Truth: [1 1 0] Predicted: [1 1 0] Overlap: [1 1 0] Union: [1 1 0]
    [1 1 0] [1 0 0] [1 0 0] [1 1 0]
    [0 0 0] [0 0 0] [0 0 0] [0 0 0]
    IoU = 3 / 4 = 0.75
  - Explanation:
    - Overlap: Pixels where both ground truth and prediction are 1
    - Union: Pixels where either ground truth or prediction is 1
    - IoU = Number of overlapping pixels / Number of pixels in union

Segmentation Evaluation metrics

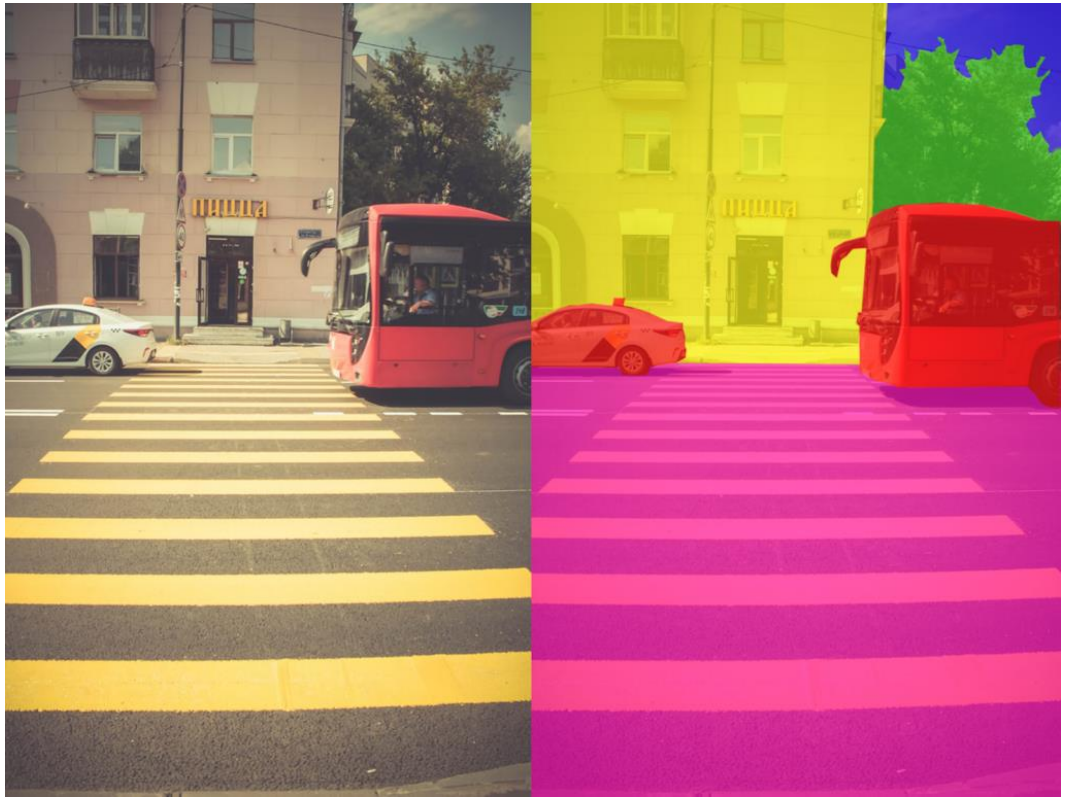
- mIoU (mean Intersection over Union): Average IoU across all classes in a segmentation task
  - For a 3-class segmentation (background, cat, dog):
    - Background IoU = 0.9
    - Cat IoU = 0.7
    - Dog IoU = 0.8
  - Then:
    - mIoU = (0.9 + 0.7 + 0.8) / 3 = 0.8

Segmentation Tasks

- Image segmentation: partitioning an image into multiple segments or regions
- Types of Segmentation:
  - Semantic Segmentation
  - Instance Segmentation
  - Panoptic Segmentation

Semantic Segmentation Task

- Semantic segmentation: assign a class label to each pixel
  - Does not distinguish between different instances of the same class
    - Example: All cars are labeled as "car" regardless of how many there are
  - CNN approaches:
    - FCN (Fully Convolutional Networks) (2015)
    - U-NET (2015)
    - DeepLab (2017)



https://raw.githubusercontent.com/oseledets/dl2024/8d739d06892c9cc71f198c1c478d51b4a6801866/lectures/lecture-4/semantic-seg.png

Semantic Segmentation Task

- FCN (Fully Convolutional Networks) (2015)
  - Key Innovations:
    - First end-to-end trainable CNN for semantic segmentation
    - Replaces fully connected layers with convolutional layers
    - Introduces skip connections to combine coarse and fine features
    - First to achieve pixel-wise prediction
  - Architecture:
    - VGG-16 backbone with fully convolutional layers
    - Skip connections from early layers
    - Transposed convolution for upsampling
  - Training: Uses pixel-wise cross-entropy loss
  - https://arxiv.org/abs/1411.4038

Semantic Segmentation Task

- FCN (Fully Convolutional Networks) (2015)

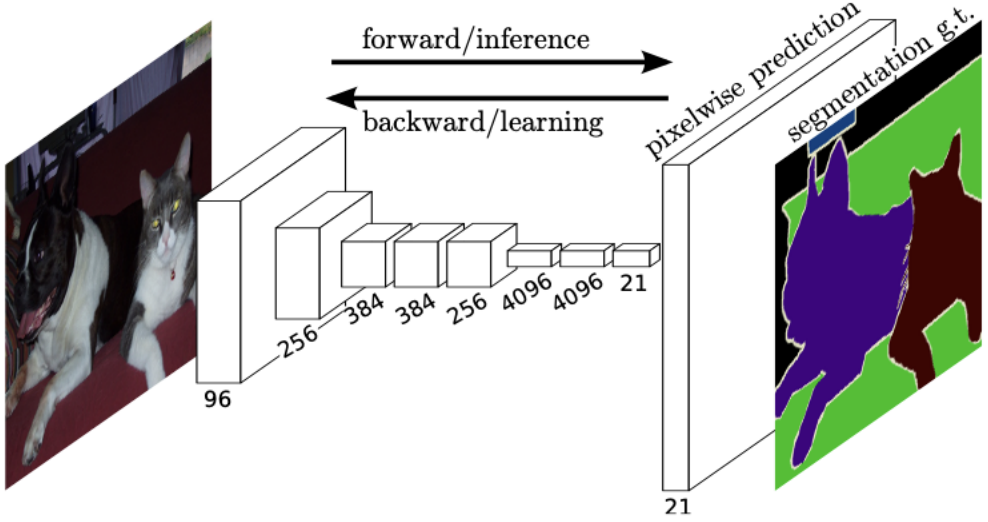


Figure 1. Fully convolutional networks can efficiently learn to make dense predictions for per-pixel tasks like semantic segmentation.

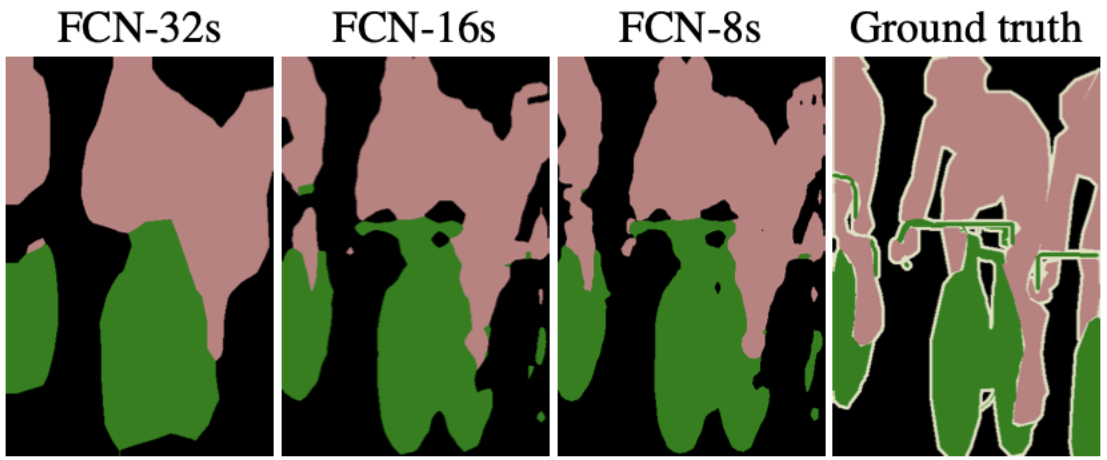


Figure 4. Refining fully convolutional nets by fusing information from layers with different strides improves segmentation detail. The first three images show the output from our 32, 16, and 8 pixel stride nets (see Figure 3).

https://arxiv.org/abs/1411.4038

Semantic Segmentation Task

- U-Net (2015)
  - Key Innovations:
    - Symmetric encoder-decoder structure
    - Skip connections between encoder and decoder
    - Better handling of fine details
    - Works well with small datasets
  - Architecture:
    - Contracting path (encoder) captures context
    - Expanding path (decoder) enables precise localization
    - Skip connections combine local and global information
  - Training: Uses weighted cross-entropy loss for better boundary detection
  - https://arxiv.org/abs/1505.04597

Semantic Segmentation Task

- U-Net (2015)

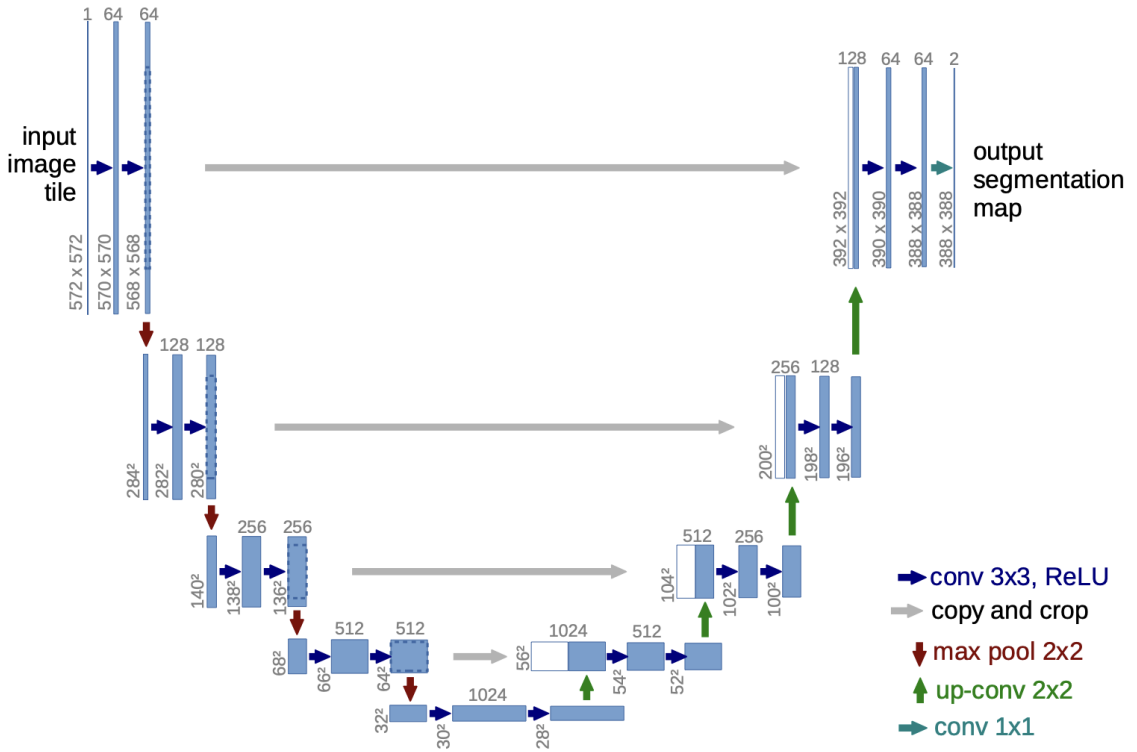


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

https://arxiv.org/abs/1505.04597

Semantic Segmentation Task

- DeepLab (2017)
  - Key Innovations:
    - Introduces atrous convolution for larger receptive field
    - Atrous spatial pyramid pooling (ASPP) module captures multi-scale context
    - Better handling of objects at different scales
    - Improved boundary accuracy
  - Architecture:
    - ResNet backbone with atrous convolution
    - ASPP module with multiple dilation rates
    - Conditional Random Field (CRF) post-processing for refinement
  - Training: Uses cross-entropy loss with auxiliary loss
  - https://arxiv.org/abs/1606.00915

Semantic Segmentation Task

- DeepLab (2017)

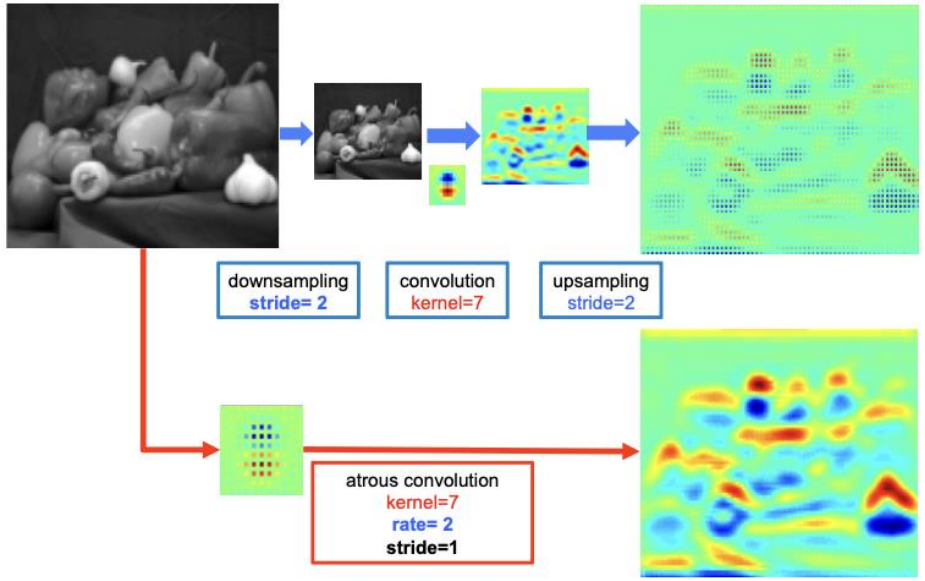


Fig. 3: Illustration of atrous convolution in 2-D. Top row: sparse feature extraction with standard convolution on a low resolution input feature map. Bottom row: Dense feature extraction with atrous convolution with rate $r = 2$, applied on a high resolution input feature map.

Method	MSC	COCO	Aug	LargeFOV	ASPP	CRF	mIOU
VGG-16							
DeepLab [38]				✓			37.6
DeepLab [38]				✓		✓	39.6
ResNet-101							
DeepLab							39.6
DeepLab	✓		✓				41.4
DeepLab	✓	✓	✓				42.9
DeepLab	✓	✓	✓	✓			43.5
DeepLab	✓	✓	✓		✓		44.7
DeepLab	✓	✓	✓		✓	✓	45.7
O_2P [45]							18.1
CFM [51]							34.4
FCN-8s [14]							37.8
CRF-RNN [59]							39.3
ParseNet [86]							40.4
BoxSup [60]							40.5
HO_CRF [91]							41.3
Context [40]							43.3
VeryDeep [93]							44.5

TABLE 6: Comparison with other state-of-art methods on PASCAL-Context dataset.

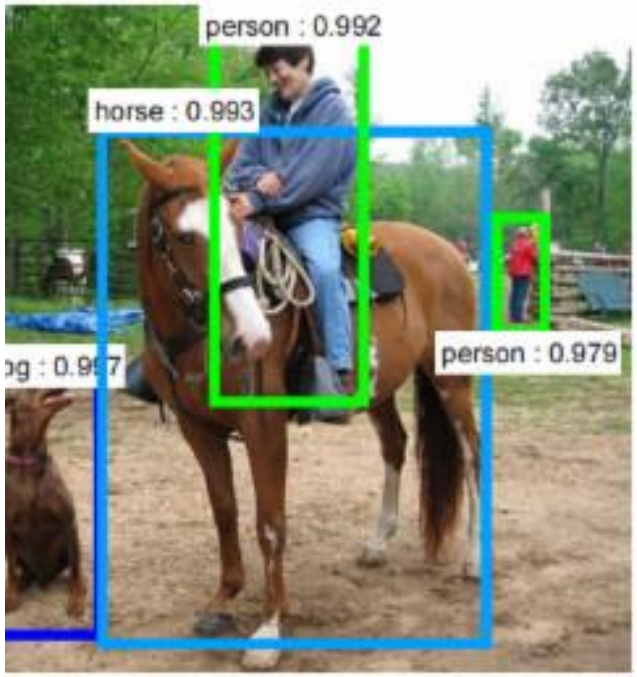
https://arxiv.org/abs/1606.00915

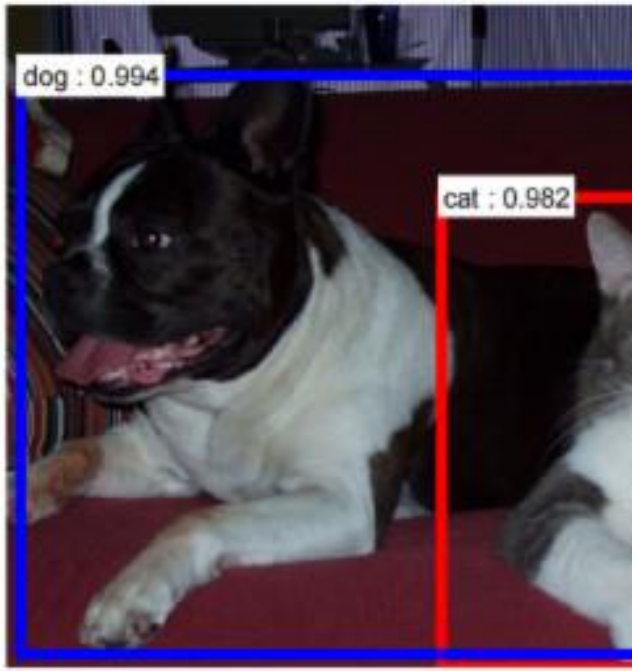
Semantic Segmentation Task

- Datasets:
  - PASCAL VOC (Visual Object Classes)
    - 20 object classes, pixel-level annotations, includes object boundaries
    - http://host.robots.ox.ac.uk/pascal/VOC/
  - Cityscapes
    - Urban street scenes, 30 classes, fine and coarse annotations, includes stereo images
    - https://www.cityscapes-dataset.com/
  - ADE20K
    - 20,000 images, 150 object categories, scene parsing annotations
    - https://groups.csail.mit.edu/vision/datasets/ADE20K/

Bounding Box Detection Task

- Bounding box detection: find objects by means of a rectangular bounding box with a class
- CNN approaches:
  - R-CNN Family (2014-2017)
  - YOLO (2016)
  - SSD (2016)





https://arxiv.org/abs/1506.01497

Bounding Box Detection Task

- R-CNN (Region-based CNN) (2014)
  - Key Innovations:
    - First CNN-based object detector
    - Uses selective search for region proposals
    - Extracts CNN features for each region
    - Uses SVM for classification
    - Linear regression for bounding box refinement
  - Architecture:
    - Region proposal generation (selective search)
    - CNN feature extraction (AlexNet)
    - SVM classification
    - Bounding box regression
  - Limitations:
    - Slow (processes each region separately)
    - Multi-stage training
  - https://arxiv.org/abs/1311.2524

Bounding Box Detection Task

- Fast R-CNN (2015)
  - Key Innovations:
    - Processes entire image through CNN once
    - Uses RoI (Region of Interest) pooling to extract features
    - End-to-end training
    - Multi-task loss (classification + regression)
  - Architecture:
    - CNN backbone
    - RoI pooling layer
    - Classification and regression heads
  - Improvements:
    - Faster training and inference
    - Better accuracy
    - Single-stage training
  - https://arxiv.org/abs/1504.08083

Bounding Box Detection Task

- Faster R-CNN (2016)
  - Key Innovations:
    - Region Proposal Network (RPN)
    - Anchor boxes
    - Shared features between RPN and detection
  - Architecture:
    - CNN backbone
    - RPN for region proposals
    - RoI pooling
    - Classification and regression heads
  - Advantages:
    - Real-time detection
    - Better accuracy
    - Fully convolutional
  - https://arxiv.org/abs/1506.01497

Bounding Box Detection Task

- Faster R-CNN (2016)

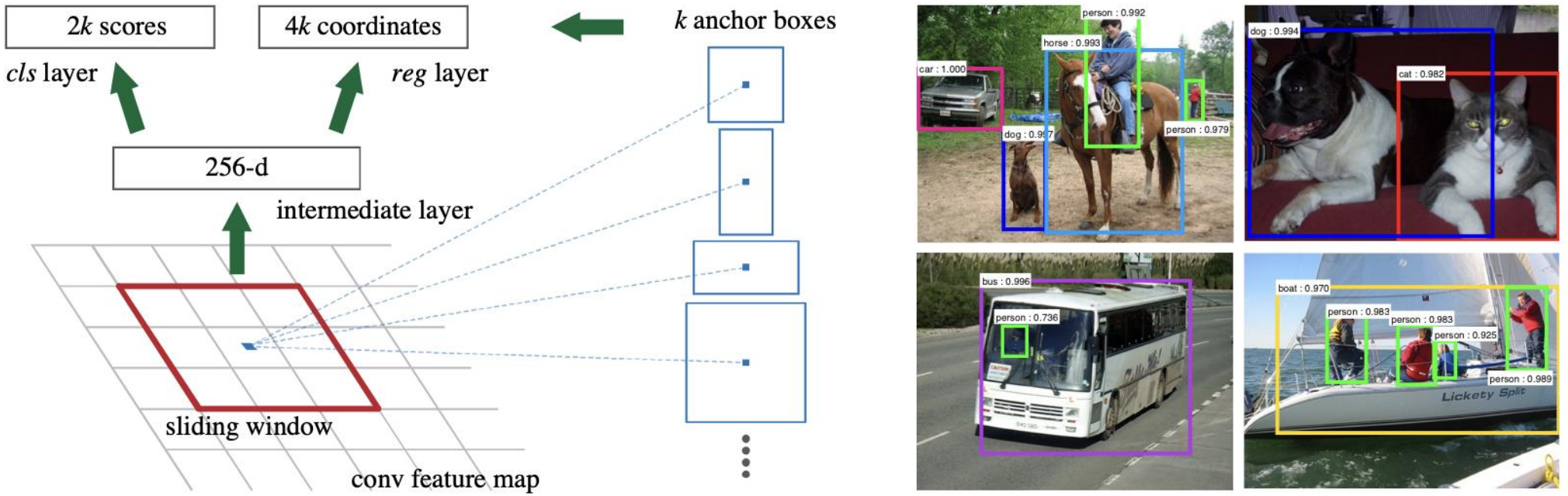


Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

https://arxiv.org/abs/1506.01497

Bounding Box Detection Task

- YOLOv1 (2016)
  - Key Innovations:
    - Divides image into grid cells
    - Each cell predicts bounding boxes and classes
    - End-to-end training
  - Architecture:
    - 24 convolutional layers
    - 2 fully connected layers
    - Grid-based prediction
  - Advantages:
    - Very fast
    - Global context
    - Simple architecture
  - https://arxiv.org/abs/1506.02640

Bounding Box Detection Task

- YOLOv1 (You Only Look Once) (2016)

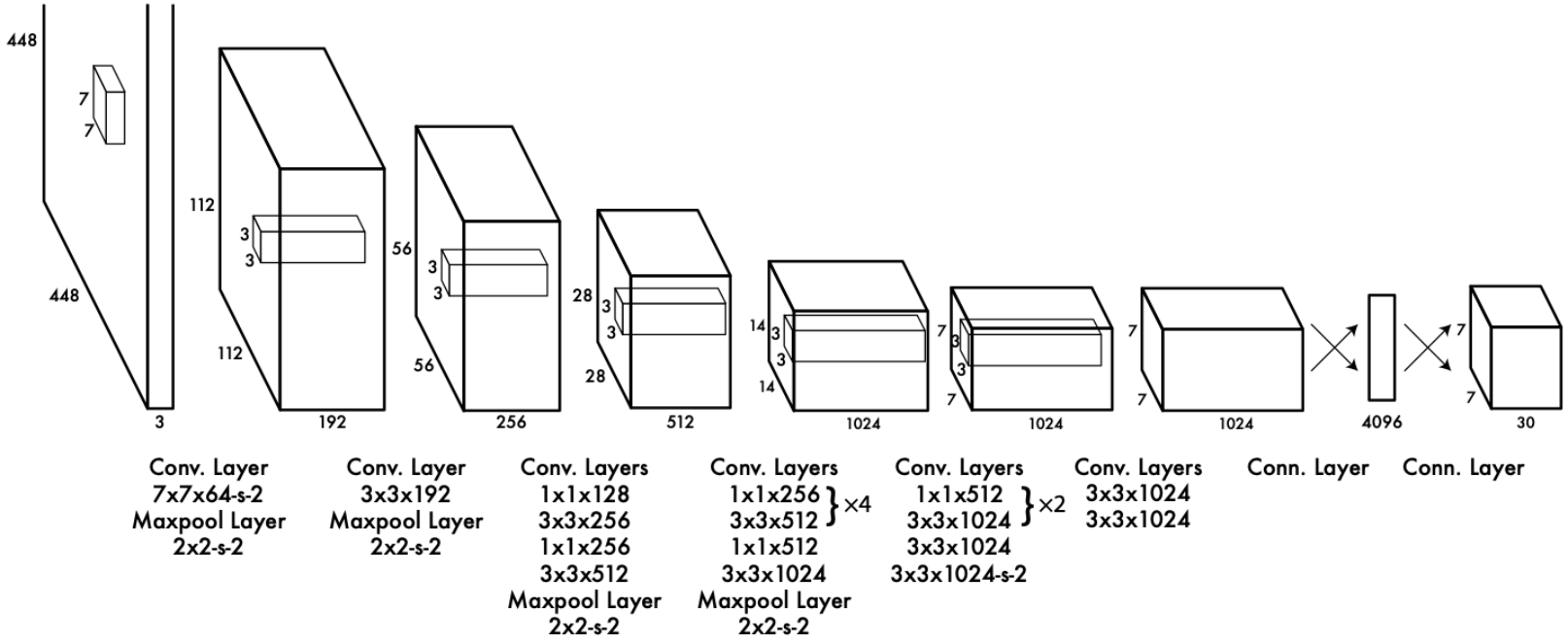


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

https://arxiv.org/abs/1506.02640

Bounding Box Detection Task

- YOLOv1 (2016)

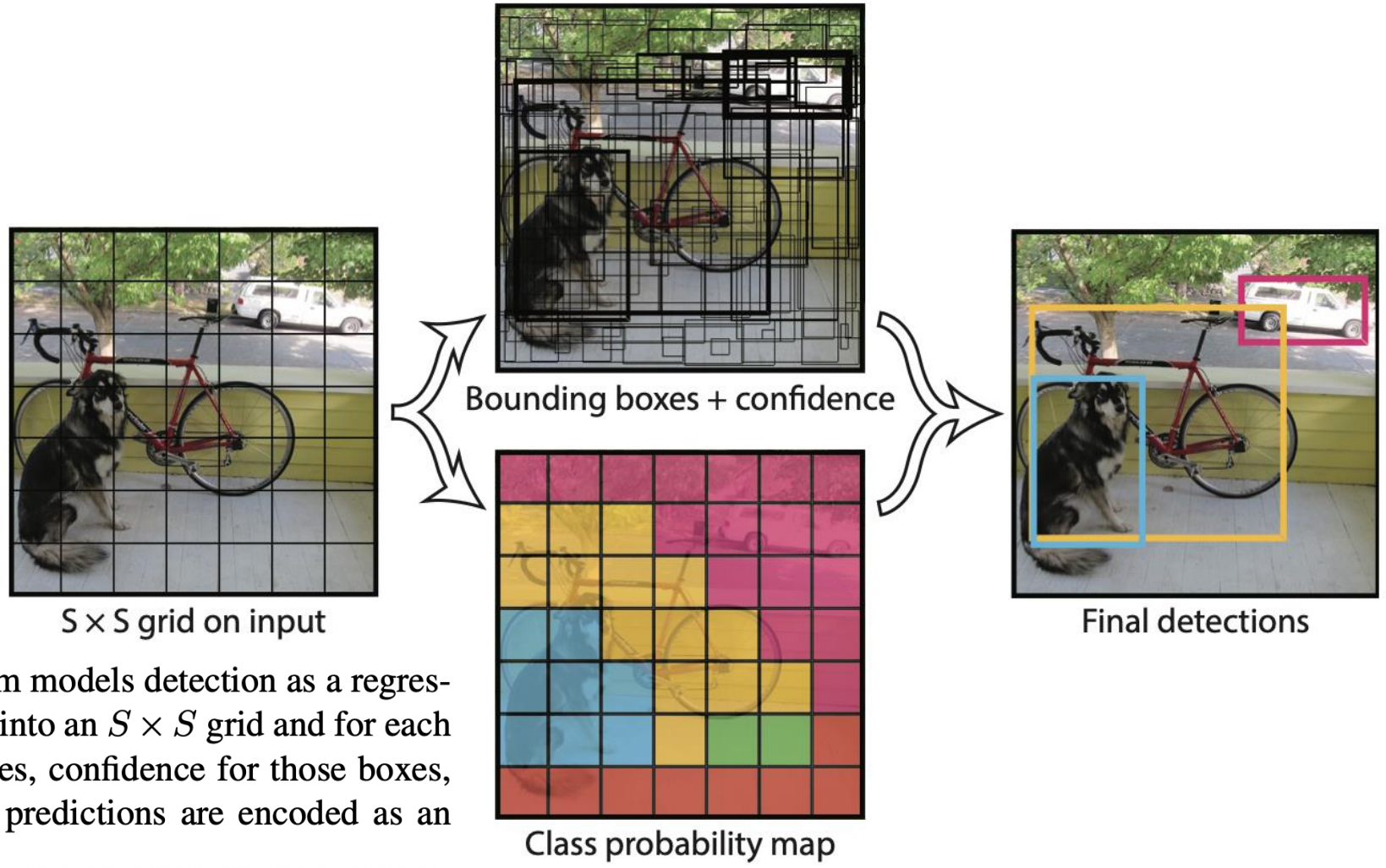


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

https://arxiv.org/abs/1506.02640

Bounding Box Detection Task

- YOLOv2/YOLO9000 (2017)
  - Key Innovations:
    - Batch normalization
    - High resolution classifier
    - Anchor boxes
    - Multi-scale training
  - Architecture:
    - Darknet-19 backbone
    - Anchor box prediction
  - Improvements:
    - Better accuracy
    - Faster than YOLOv1
    - Detects 9000 classes
  - https://arxiv.org/abs/1612.08242

Bounding Box Detection Task

- YOLOv3 (2018)
  - Key Innovations:
    - Feature pyramid network (we will see it in Instance Segmentation Task)
    - Better backbone (Darknet-53)
    - Binary cross-entropy loss
  - Architecture:
    - Darknet-53 backbone
    - Three detection scales
    - Feature pyramid network
  - Advantages:
    - Better small object detection
    - Improved accuracy
    - Good speed-accuracy trade-off
  - https://arxiv.org/abs/1804.02767

Bounding Box Detection Task

- YOLOv4 (2020)
  - Key Innovations:
    - CSPDarknet53 backbone
    - PANet feature aggregation
    - Mosaic data augmentation
    - CIoU loss (to improve imbalanced datasets)
  - Architecture:
    - CSPDarknet53 backbone
    - SPP and PANet modules
    - YOLOv3 head
  - Improvements:
    - Better accuracy
    - Faster training
    - More robust
  - https://arxiv.org/abs/2004.10934

Bounding Box Detection Task

- SSD (Single Shot MultiBox Detector) (2016)
  - Key Innovations:
    - Multi-scale feature maps
    - Default boxes (similar to anchor boxes)
    - Hard negative mining (takes into account negative samples incorrectly identified)
    - Data augmentation
  - Architecture:
    - VGG-16 backbone
    - Additional convolutional layers
    - Multi-scale prediction layers
  - Advantages:
    - Good speed-accuracy trade-off
    - Simple architecture
    - Easy to implement
  - https://arxiv.org/abs/1512.02325

Bounding Box Detection Task

- SSD (2016)

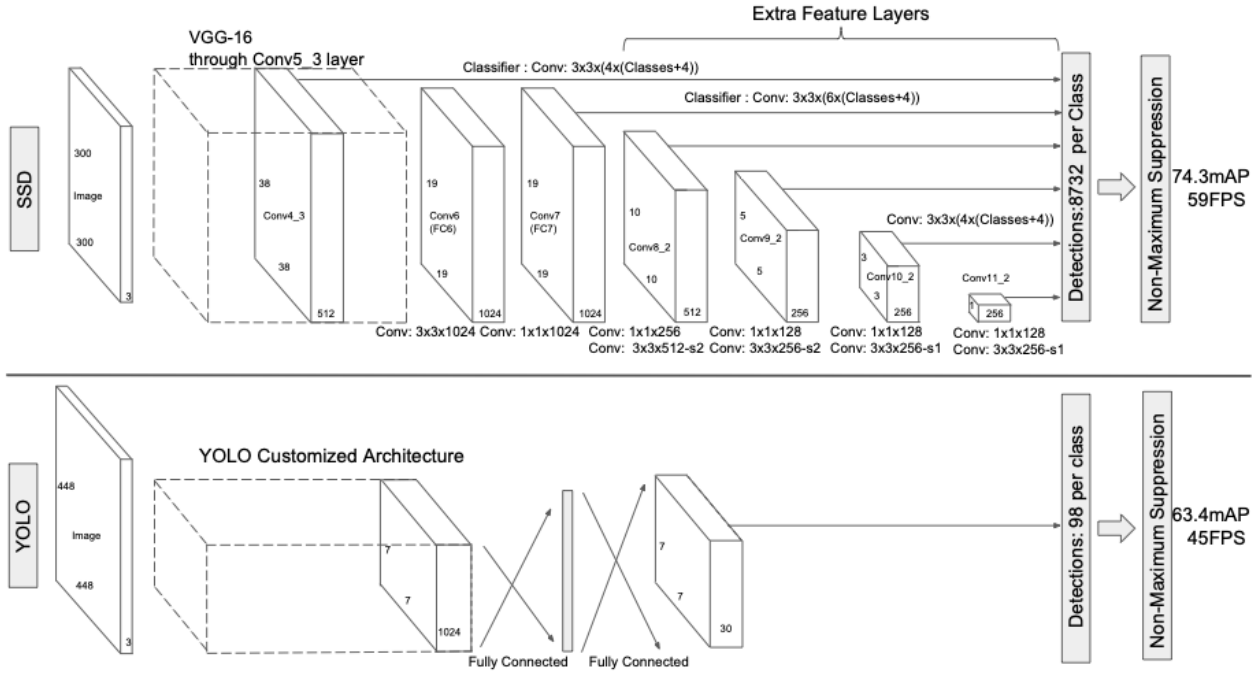
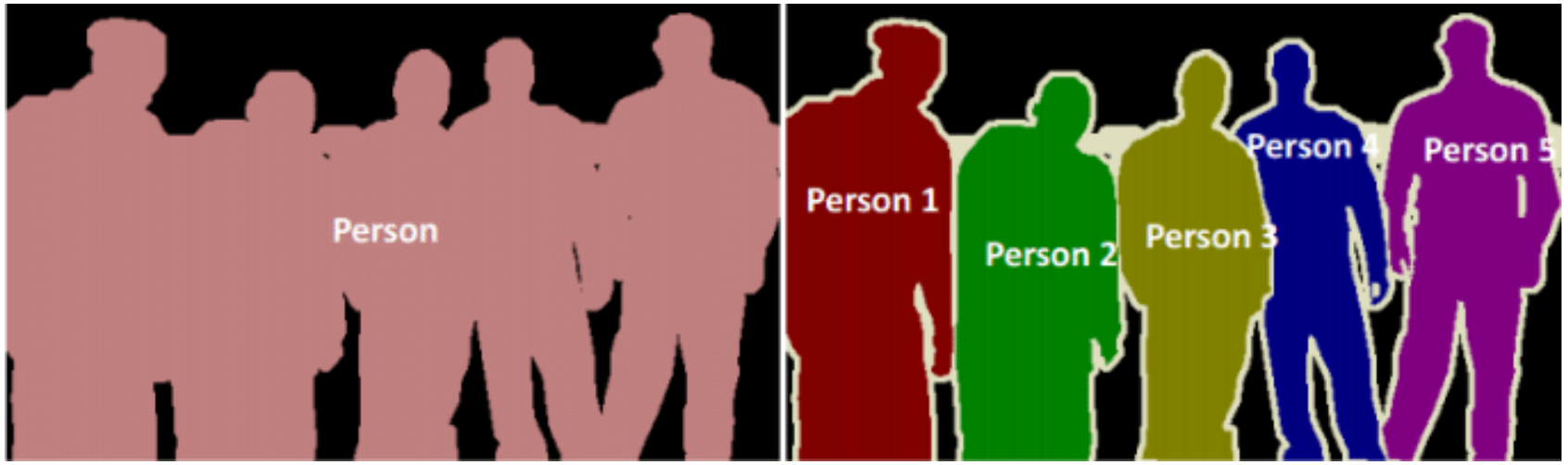


Fig. 2: A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300×300 input size significantly outperforms its 448×448 YOLO counterpart in accuracy on VOC2007 `test` while also improving the speed.

https://arxiv.org/abs/1512.02325

Instance Segmentation Task

- Instance segmentation: identifies and segments each object instance separately
  - Distinguishes between different instances of the same class
    - Each car gets a unique label (car1, car2, etc.)
    - Each person gets a unique label (person1, person2, etc.)



https://raw.githubusercontent.com/oseledets/dl2024/8d739d06892c9cc71f198c1c478d51b4a6801866/lectures/lecture-4/instance.png

Instance Segmentation Task

- Instance segmentation: identifies and segments each object instance separately
  - Distinguishes between different instances of the same class
    - Each car gets a unique label (car1, car2, etc.)
    - Each person gets a unique label (person1, person2, etc.)
  - CNN approaches:
    - Mask R-CNN (2017)
    - YOLACT (2019)

Instance Segmentation Task

- Mask R-CNN (2017)
  - Approach: Extends Faster R-CNN with mask prediction branch
  - Key Innovations:
    - Adds parallel mask branch to object detection
    - Uses RoIAlign (RoI for Region of Interest) layer for better mask accuracy
    - Handles both detection and segmentation
  - Architecture:
    - ResNet and FPN (Feature Pyramid Network) backbones
    - Region Proposal Network (RPN)
    - Mask prediction branch
  - Training: Multi-task loss (classification, box regression, mask)
  - https://arxiv.org/abs/1703.06870

Instance Segmentation Task

- Mask R-CNN (2017)

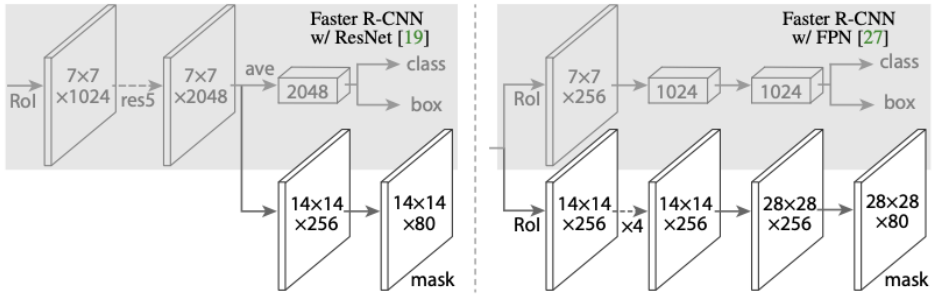


Figure 4. **Head Architecture**: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively, to which a mask branch is added. Numbers denote spatial resolution and channels. Arrows denote either conv, deconv, or *fc* layers as can be inferred from context (conv preserves spatial dimension while deconv increases it). All convs are 3×3, except the output conv which is 1×1, deconvs are 2×2 with stride 2, and we use ReLU [31] in hidden layers. *Left*: 'res5' denotes ResNet's fifth stage, which for simplicity we altered so that the first conv operates on a 7×7 RoI with stride 1 (instead of 14×14 / stride 2 as in [19]). *Right*: '×4' denotes a stack of four consecutive convs.

description	backbone	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}	AP^{kp}_{M}	AP^{kp}_{L}
original baseline	R-50-FPN	64.2	86.6	69.7	58.7	73.0
+ updated baseline	R-50-FPN	65.1	86.6	70.9	59.9	73.6
+ deeper	R-101-FPN	66.1	87.7	71.7	60.5	75.0
+ ResNeXt	X-101-FPN	67.3	88.0	73.3	62.2	75.6
+ data distillation [35]	X-101-FPN	**69.1**	**88.9**	**75.3**	**64.1**	**77.1**
+ test-time augm.	X-101-FPN	**70.4**	**89.3**	**76.8**	**65.8**	**78.1**

Table 9. **Enhanced keypoint results** of Mask R-CNN on COCO `minival`. Each row adds an extra component to the above row. Here we use only keypoint annotations but no mask annotations. We denote ResNet by 'R' and ResNeXt by 'X' for brevity.

https://arxiv.org/abs/1703.06870

Instance Segmentation Task

- YOLACT (2019)
  - Key Innovations:
    - First real-time instance segmentation model
    - Uses prototype masks and mask coefficients
    - Fast inference speed
  - Architecture:
    - Any convolutional backbone network
    - Prototype generation branch
    - Mask coefficient prediction
  - Training: Uses combination of classification and mask losses
  - https://arxiv.org/abs/1904.02689

Instance Segmentation Task

- YOLACT (2019)

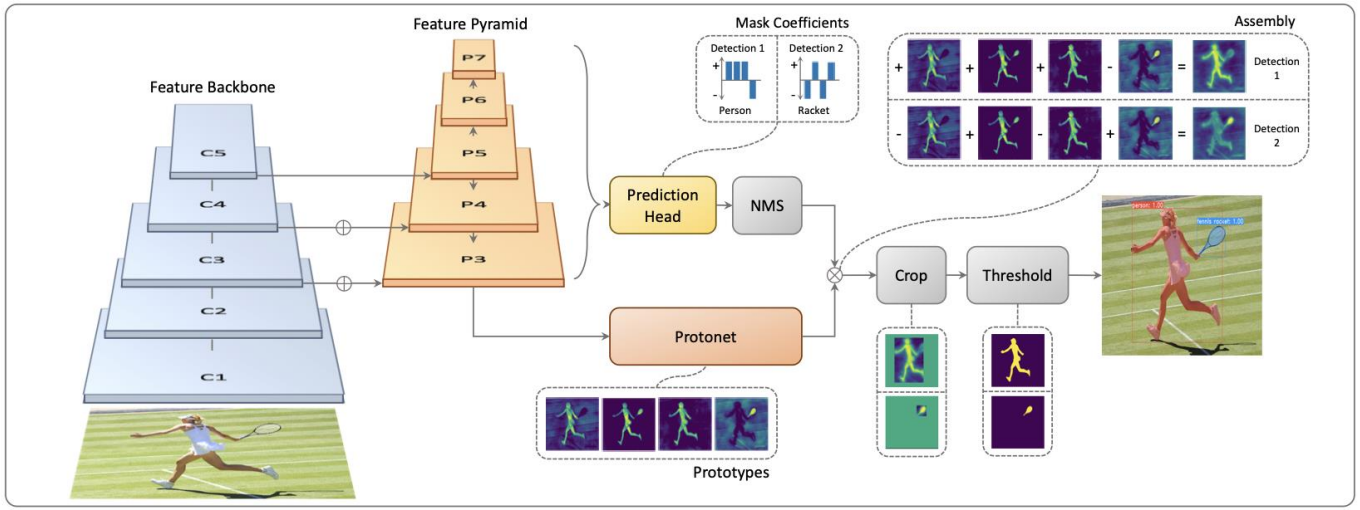


Figure 2: **YOLACT Architecture** Blue/yellow indicates low/high values in the prototypes, gray nodes indicate functions that are not trained, and $k = 4$ in this example. We base this architecture off of RetinaNet [27] using ResNet-101 + FPN.

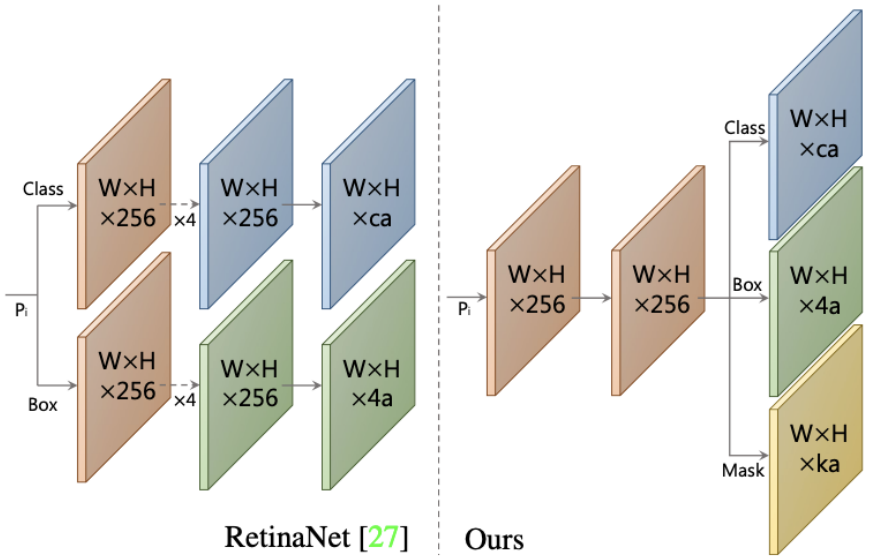


Figure 4: **Head Architecture** We use a shallower prediction head than RetinaNet [27] and add a mask coefficient branch. This is for c classes, a anchors for feature layer P_i, and k prototypes. See Figure 3 for a key.

https://arxiv.org/abs/1904.02689

Instance Segmentation Task

- Datasets:
  - COCO (Common Objects in Context)
    - 330,000 images
    - 80 object categories
    - Instance-level annotations
    - https://cocodataset.org/
  - LVIS (Large Vocabulary Instance Segmentation)
    - 1200 categories
    - Long-tailed distribution
    - High-quality instance annotations
    - https://www.lvisdataset.org/

Panoptic Segmentation Task

- Panoptic segmentation: each pixel in a scene is assigned a semantic label and a unique instance identifier
- Combines semantic and instance segmentation
- CNN approaches:
  - Panoptic FPN (2019)
  - UPSNet (Unified Panoptic Segmentation Network) (2019)

Panoptic Segmentation Task

- Panoptic FPN (2019)
  - Key Innovations:
    - Unified architecture for both things (instance) and stuff (semantic)
    - Shared backbone for efficiency
    - Better handling of overlapping instances
  - Architecture:
    - FPN (Feature Pyramid Network) backbone
    - Semantic segmentation branch
    - Instance segmentation branch
  - Training: uses panoptic quality metric
  - https://arxiv.org/abs/1901.02446

Panoptic Segmentation Task

- Panoptic FPN (2019)

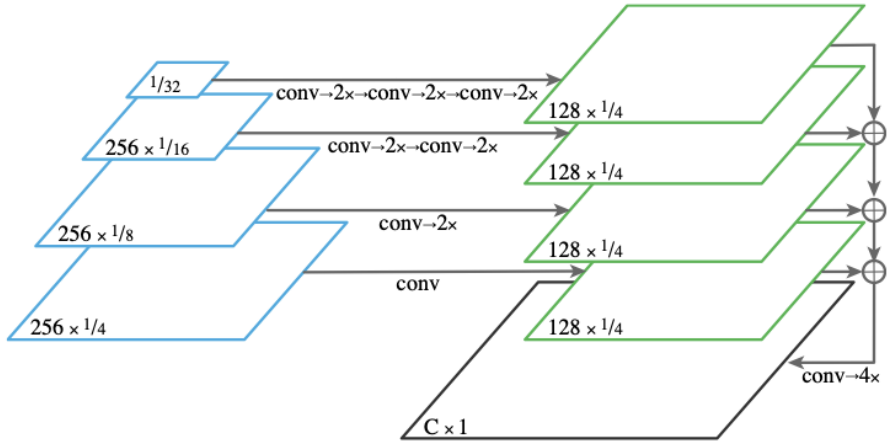


Figure 3: **Semantic segmentation branch.** Each FPN level (left) is upsampled by convolutions and bilinear upsampling until it reaches 1/4 scale (right), theses outputs are then summed and finally transformed into a pixel-wise output.

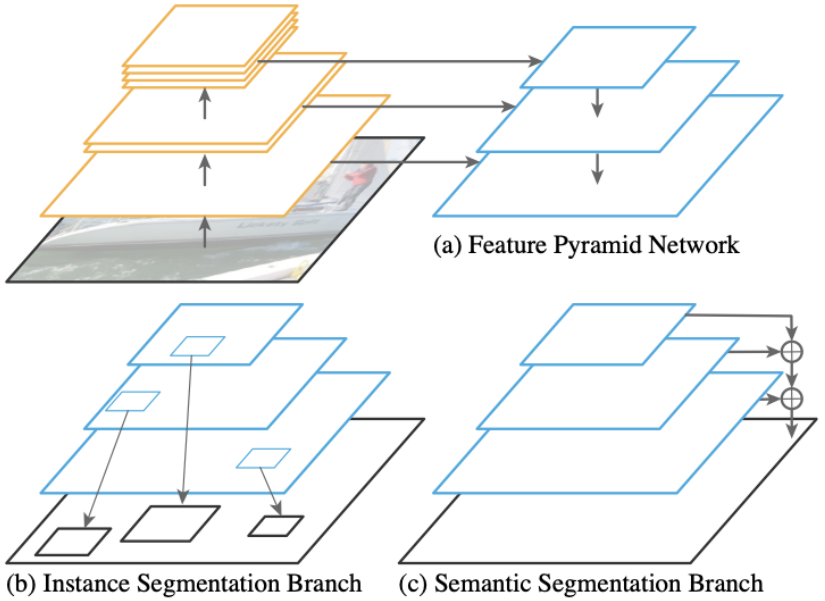


Figure 1: **Panoptic FPN**: (a) We start with an FPN backbone [36], widely used in object detection, for extracting rich multi-scale features. (b) As in Mask R-CNN [24], we use a region-based branch on top of FPN for instance segmentation. (c) In parallel, we add a lightweight dense-prediction branch on top of the same FPN features for semantic segmentation. This simple extension of Mask R-CNN with FPN is a fast and accurate baseline for both tasks.

https://arxiv.org/abs/1901.02446

Panoptic Segmentation Task

- UPSNet (Unified Panoptic Segmentation Network) (2019)
  - Key Innovations:
    - Better handling of overlapping regions
    - Improved instance segmentation
  - Architecture:
    - Shared feature extractor
    - Semantic head
    - Instance head
    - Panoptic head
  - Training: Uses panoptic segmentation loss
  - https://arxiv.org/abs/1901.03784

Panoptic Segmentation Task

- UPSNet (Unified Panoptic Segmentation Network) (2019)

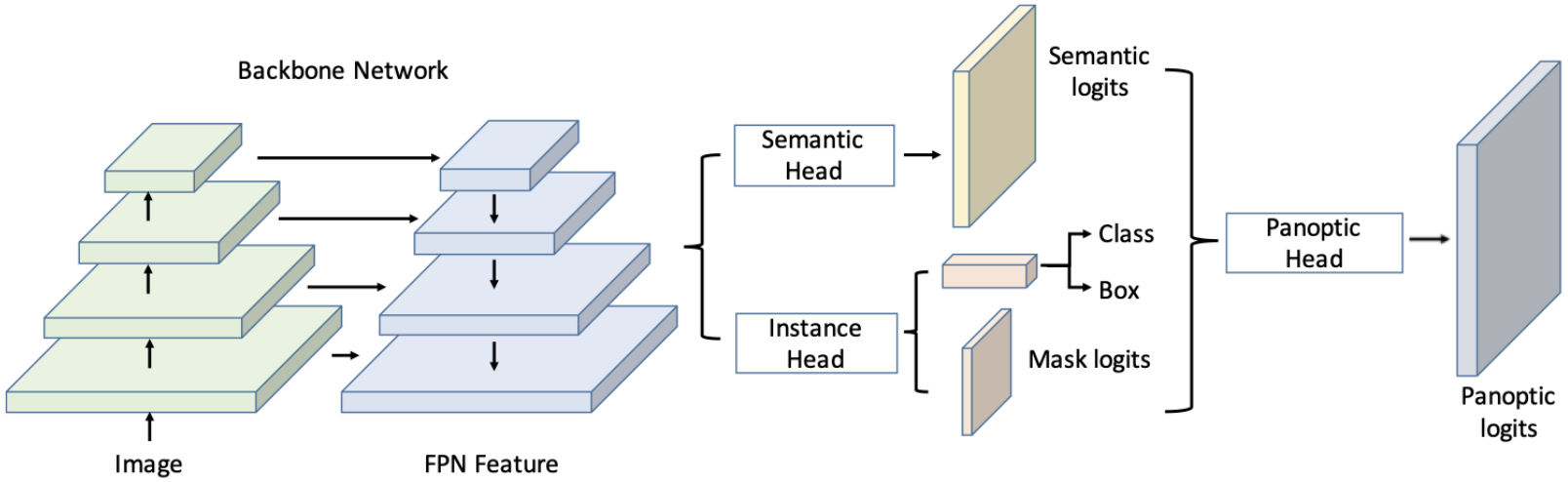


Figure 1: Overall architecture of our UPSNet.

https://arxiv.org/abs/1901.03784

Panoptic Segmentation Task

- Datasets:
  - COCO Panoptic
    - Extension of COCO dataset
    - Panoptic annotations
    - Both things and stuff categories
    - https://cocodataset.org/#panoptic-2019
  - Mapillary Vistas
    - 25,000 street-level images
    - 124 object categories
    - Panoptic annotations
    - https://www.mapillary.com/dataset/vistas