Foundations of Deep Learning, Winter Term 2021/22

Week 8: Practical Methodology and Architectures

Practical Methodology and Architectures

Frank Hutter Abhinav Valada

University of Freiburg

Overview of Week 8

1. Normalization Layers
2. Transfer Learning
3. General Methodology
4. Parameter Initialization
5. CNN Architectures
6. Detection and Segmentation
7. Summary, Further Reading, References

Lecture Overview

1. Normalization Layers
2. Transfer Learning
3. General Methodology
4. Parameter Initialization
5. CNN Architectures
6. Detection and Segmentation
7. Summary, Further Reading, References

Foundations of Deep Learning, Winter Term 2021/22

Week 8: Practical Methodology and Architectures

Normalization Layers

Frank Hutter Abhinav Valada

University of Freiburg

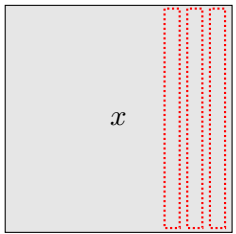
Batch Normalization

Consider a batch of activations at some layer. To make each dimension zero-mean and unit-variance, apply:

$$\widehat{x}^{(k)} = \frac{x^{(k)} - \mathsf{E}[x^{(k)}]}{\sqrt{\mathrm{Var}[x^{(k)}]}}$$

Batch Normalization (cont.)

Consider an input x of shape $N \times D$

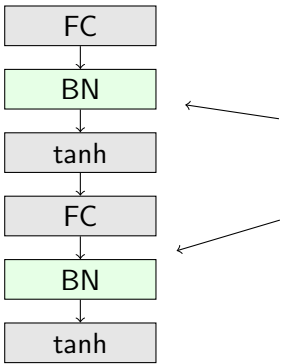


$$\mu_j \leftarrow \frac{1}{N}\sum_{i=1}^{N} x_{i,j}$$ Per-channel mean, shape D

$$\sigma_j^2 \leftarrow \frac{1}{N}\sum_{i=1}^{N}(x_{i,j} - \mu_j)^2$$ Per-channel variance, shape D

$$\widehat{x}_{i,j} \leftarrow \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}}$$ Normalized x, shape $N \times D$

Batch Normalization (cont.)



Usually inserted after fully connected or convolutional layers, but before nonlinearity.

Batch Normalization (cont.)

Consider an input x of shape $N \times D$

- Learnable scale and shift parameters: γ, β of shape D.
- Learning $\gamma_j = \sigma_j$ and $\beta_j = \mu_j$ recovers the identity function.

$$y_{i,j} \leftarrow \frac{x_{i,j} - \mu_j}{\sqrt{\sigma_j^2 + \varepsilon}} \cdot \gamma - j + \beta_j \quad \text{Output, shape } N \times D$$

$\cdot$ denotes the element-wise multiplication

Batch Normalization at Test Time

- During testing batch norm becomes a linear operator.
- It can be fused with the previous fully-connected or convolution layer.

$\mu_j \leftarrow$ Moving average over the μ_j's seen during training.
$\sigma_j \leftarrow$ Moving average over the σ_j's seen during training.

Batch Normalization for CNNs

Batch normalization for **fully-connected** layers (`torch.nn.BatchNorm1d`).

$$x : N \times D$$

Normalize

$$\mu, \sigma : 1 \times D$$
$$\gamma, \beta : 1 \times D$$

$$y \leftarrow \gamma \widehat{x} + \beta$$

Batch normalization for **convolutional** layers (`torch.nn.BatchNorm2d`).

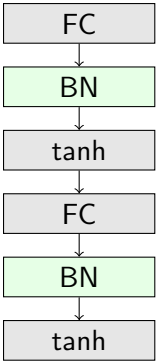
$$x : N \times C \times H \times W$$

Normalize

$$\mu, \sigma : 1 \times C \times 1 \times 1$$
$$\gamma, \beta : 1 \times C \times 1 \times 1$$

$$y \leftarrow \gamma \widehat{x} + \beta$$

Batch Normalization for CNNs (cont.)



- Makes deep networks much easier to train.
- Improves gradient flow.
- Allows higher learning rates, faster convergence.
- Networks become more robust to initialization.
- Acts as regularization during training.
- Does not perform well with small mini-batch sizes and while having recurrent connections.

Layer Normalization

Batch normalization for fully-connected layers

$$x : N \times D$$

Normalize

$$\mu, \sigma : 1 \times D$$
$$\gamma, \beta : 1 \times D$$

$$y \leftarrow \gamma \widehat{x} + \beta$$

Layer normalization for fully-connected layers behaves the same during training and testing. It can be used in RNNs and transformers.

$$x : N \times D$$

Normalize

$$\mu, \sigma : N \times 1$$
$$\gamma, \beta : 1 \times D$$

$$y \leftarrow \gamma \widehat{x} + \beta$$

Instance Normalization

Batch Normalization for convolutional layers.

$$x : N \times C \times H \times W$$

Normalize

$$\mu, \sigma : 1 \times C \times 1 \times 1$$
$$\gamma, \beta : 1 \times C \times 1 \times 1$$

$$y \leftarrow \gamma \widehat{x} + \beta$$

Instance Normalization for convolutional layers. Same behavior for train and test.

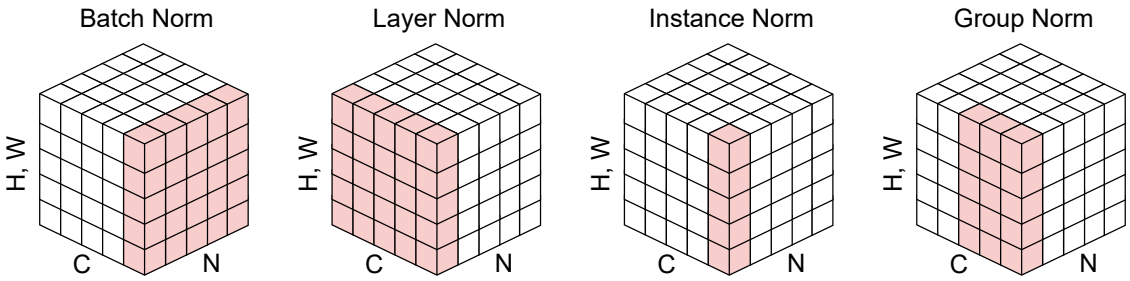
$$x : N \times C \times H \times W$$

Normalize

$$\mu, \sigma : N \times C \times 1 \times 1$$
$$\gamma, \beta : 1 \times C \times 1 \times 1$$

$$y \leftarrow \gamma \widehat{x} + \beta$$

Comparison of Normalization Layers



Questions to Answer for Yourself / Discuss with Friends

- Repetition: What are two scenarios in which batch normalization does not perform well?
- Repetition: Which normalization should you use for RNNs?
- Repetition: Which normalization has been demonstrated to perform well for style transfer?

Lecture Overview

1. Normalization Layers
2. Transfer Learning
3. General Methodology
4. Parameter Initialization
5. CNN Architectures
6. Detection and Segmentation
7. Summary, Further Reading, References

Foundations of Deep Learning, Winter Term 2021/22

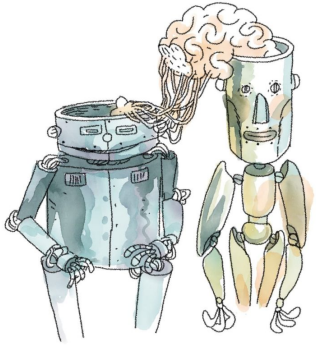
Week 8: Practical Methodology and Architectures

Transfer Learning

Frank Hutter Abhinav Valada

University of Freiburg

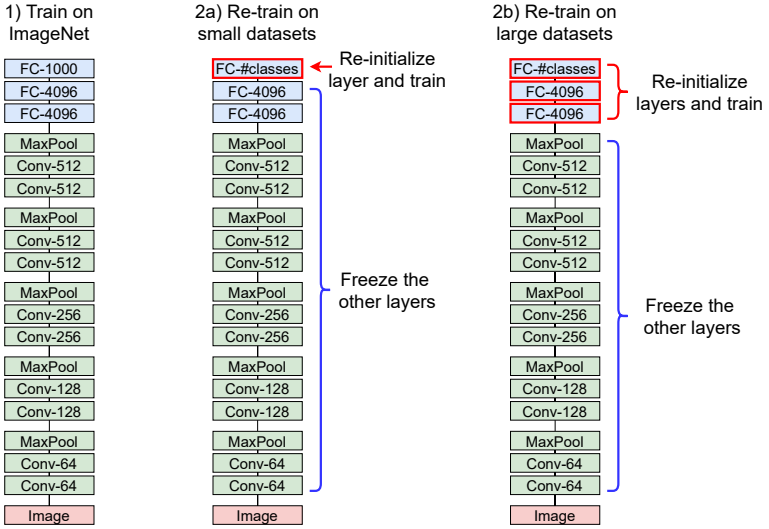
Transfer Learning



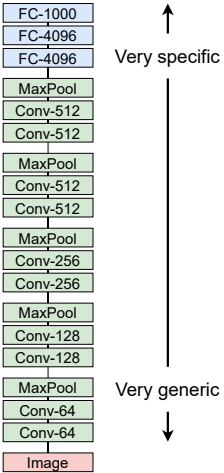
- Idea: Make use of a large related dataset.
- Initialize model for small dataset with weights found on a large dataset.

[Image source: https://thedatafrog.com/en/articles/image-recognition-transfer-learning]

Transfer Learning with CNNs



Transfer Learning with CNNs – Strategies



	Similar dataset	**Different dataset**
Little data	Add linear classifier to top layer.	Difficult to resolve. Try linear classifier at different stages.
Much data	Fine-tune some layers.	Fine-tune many layers.

Self-Supervised Learning

It can be beneficial to first train without human supervision.

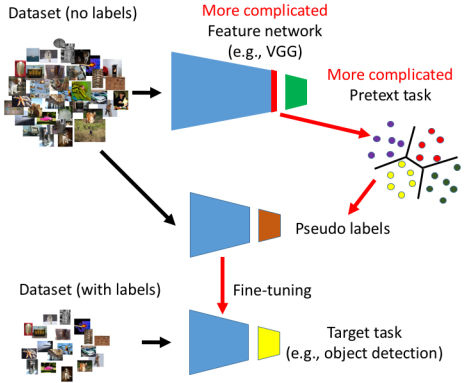


Figure: Vision [Noroozi et al., 2018].

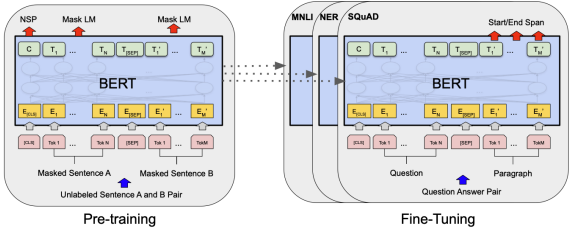


Figure: Natural language processing [Devlin et al., 2019].

Transfer Learning in Vision

- Self-supervised Learning
- Image Classification

⋮

- Image Classification
- Image Captioning
- Object Detection
- Image Segmentation

⋮

Transfer Learning in NLP

- Language Modeling
- Masked Language Modeling

⋮

- Question Answering
- Sentiment Classification
- Entailment Classification
- Grammar Checking
- Machine Translation

⋮

Questions to Answer for Yourself / Discuss with Friends

- Repetition: How will you perform transfer learning when you have very little data but your data is similar to ImageNet?
- Application of what you just learned: Can you still perform transfer learning if the network architecture that you used to first pre-train on ImageNet is completely different from the architecture that you are going to use for training on your small dataset?

Lecture Overview

1. Normalization Layers
2. Transfer Learning
3. General Methodology
4. Parameter Initialization
5. CNN Architectures
6. Detection and Segmentation
7. Summary, Further Reading, References

Foundations of Deep Learning, Winter Term 2021/22

Week 8: Practical Methodology and Architectures

General Methodology

Frank Hutter Abhinav Valada

University of Freiburg

What Deep Learning Model Should I Apply?

- Fixed-size vectors as input?
  - ⇝ Feedforward network with fully connected layers
- Input with topological structure (e.g., images)?
  - ⇝ Convolutional network
  - ⇝ Transformer
- Input (and output) is a sequence?
  - ⇝ Gated recurrent net (LSTM, GRU)
  - ⇝ Transformer

Recommended Practical Design Process

- Determine your goals, including performance metrics.
- Establish a working end-to-end pipeline as early as possible.
  - If task is similar to a previous one, use the same type of model or copy weights (e.g., ImageNet features).
- Determine the bottlenecks in performance: diagnose and debug.
- Repeatedly make incremental changes.
  - Gather new data.
  - Adjust hyperparameters.

Debugging Strategies

- Fit a tiny dataset.
- Visualize the exact inputs.
- Visualize the model in action.
- Visualize the worst mistakes.
- Check your derivatives with finite differences.
- Monitor histograms of activations and gradient.

Fit a Tiny Dataset

- If the training error does not vanish on a tiny dataset, something is wrong:
  - Some software bug.
  - Problem in the optimizer.
  - $\rightsquigarrow$ Very cheap check, very effective.

- Related: feed in random data as input.
  - Allows diagnosing whether output biases reflect class distribution.

Visualize the Exact Inputs

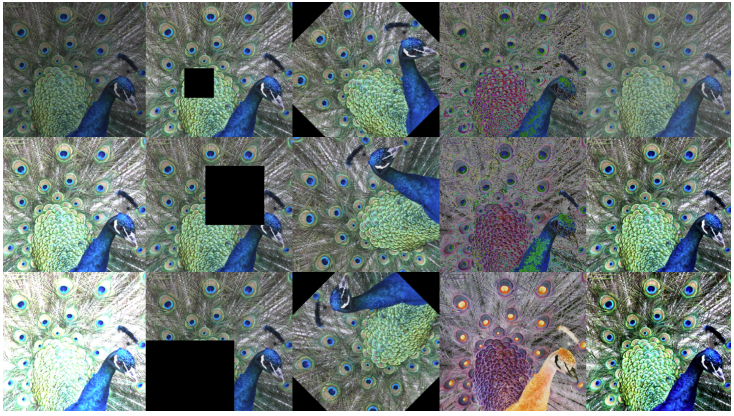
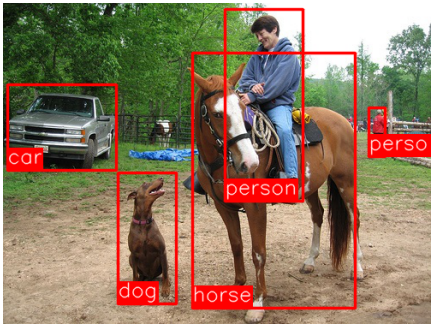


Figure: E.g., visualize the inputs of a neural network after applying augmentations.

Visualize the Model in Action



Directly observe model performance:

- Visualize the detections proposed by the model.
- Listen to speech samples if the model generates speech.
- Quantitative performance measurement, e.g., accuracy, log-likelihood, etc.

[Image source: https://cdn-images-1.medium.com/max/1600/1*E-8oQW8ZO-hHgTf6laWhhQ.png]

Visualize the Worst Mistakes

- View the training set examples that are hardest to model correctly.
- Example: Incorrectly transcribed street numbers.

[Image source: Goodfellow et al., 2013]

Check Your Derivatives with Finite Differences

- Incorrect implementation of the gradient expression is a common mistake when dealing with gradients.
- Using finite difference to verify correctness:

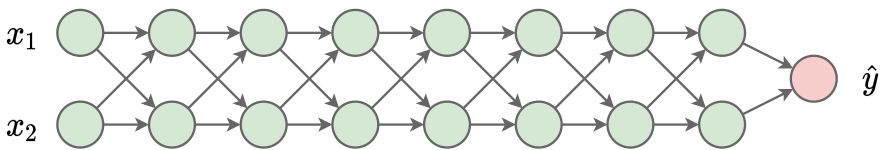
$$f'(x) = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

$$f'(x) \approx \frac{f(x+\varepsilon) - f(x)}{\varepsilon}$$

$$f'(x) \approx \frac{f\left(x+\frac{\varepsilon}{2}\right) - f\left(x-\frac{\varepsilon}{2}\right)}{\varepsilon}$$

- To check gradient on a vector-valued function $g : R^m \to R^n$,
  - simply run finite differentiation $m \times n$ times or
  - sample random directions $\mathbf{u}$ and $\mathbf{v}$ and check derivatives along those: $f(x) = \mathbf{u}^T g(\mathbf{v}x)$.

Monitor Histograms of Activations and Gradient



- Monitor gradient values to detect vanishing/exploding gradients.
- Monitor activations, e.g., how often are rectifiers off?

Questions to Answer for Yourself / Discuss with Friends

- Repetition: You have trained your network and you find that it performs really poorly in the quantitative metrics, how would you debug this?
- Repetition: You have an innovative idea for a completely new type of activation function or layer and want to implement it yourself. Which checks should you do to make sure it works as expected?

Lecture Overview

1. Normalization Layers
2. Transfer Learning
3. General Methodology
4. Parameter Initialization
5. CNN Architectures
6. Detection and Segmentation
7. Summary, Further Reading, References

Foundations of Deep Learning, Winter Term 2021/22

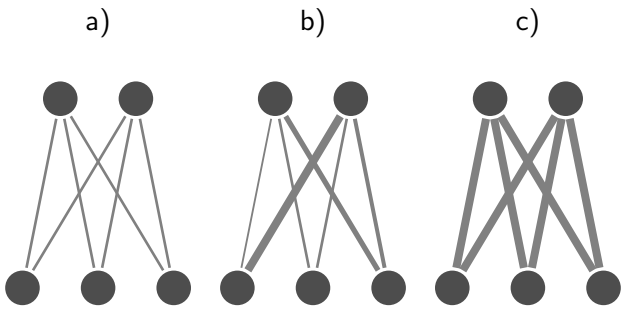
Week 8: Practical Methodology and Architectures

Parameter Initialization

Frank Hutter Abhinav Valada

University of Freiburg

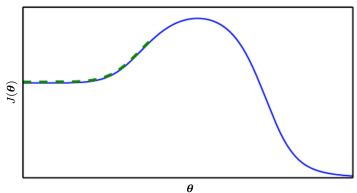
Parameter Initialization



Before we can start SGD, we need to define with what weights to start!

Parameter Initialization Strategies

- Initial point affects *convergence*, *minimal cost*, and *generalization error*.



- Current research topic, often just *simple heuristics*.
- Break the symmetry: weights of hidden units in the same layer would always remain the same if initialized the same.
  - Rather initialize weights for units to compute maximally-different functions.
- Random weights (constant bias).
- More advanced: pre-training, meta-learning.

[Image source: Goodfellow et al., 2016, p. 288, Fig. 8.4]

Parameter Initialization Strategies – Weights

- Uniform or normal distribution.
- Scale is important:
  - Break symmetry, activation flow $\rightarrow$ large
  - Gradient explosion, generalization $\rightarrow$ small
- E.g., normalized initialization (also known as Xavier initialization):

$$\mathrm{W}_{i,j} \sim U\left(-\sqrt{\frac{6}{m+n}}, \sqrt{\frac{6}{m+n}} \right)$$

  - For fully connected layer with m inputs and n outputs.
  - To compromise between goals of initializing all layers to have the same activation variance and gradient variance.
- In practice: scale of weights is a layer-dependent hyperparameter to be tuned.
- Try to preserve norm of activations & gradient when propagated through the network.
  - Can be checked and adjusted on a few mini-batches.

Parameter Initialization Strategies – Biases

- Often 0, except
  1. Output units: set bias vector $\mathbf{b}$ to match desired output statistics.
     - E.g., to output the average class proportions c_i with a softmax unit, solve $\text{softmax}(\mathbf{b}) = \mathbf{c}$.
  2. ReLU units: $\mathbf{b} \approx 0.1$ (not 0 to avoid saturation).

     Gating units: $\mathbf{b} \approx 1$ (to open the gate):
     - Gating units $h \in [0, 1]$, to be multiplied by units with output u to form uh.
     - Initialize biases to make $h \approx 1$, so that u has a chance to learn.
     - E.g., initialize the bias of the forget gate in LSTMs to 1.

Other Initialization Strategies: Meta-Learning and Warm-Starting

Both meta-learning and warm-starting techniques can be used to initialize networks.

- **Warm-starting**: initialize with weights that have been optimized on a different task.
- **Meta-learning**: learning the "best" weight initialization strategy for different tasks.

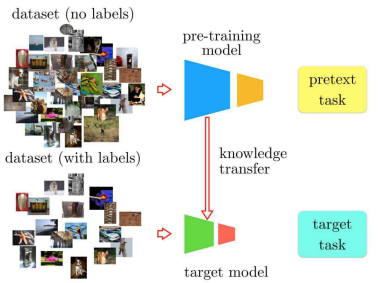


Figure: Warm-start [Noroozi et al., 2018].

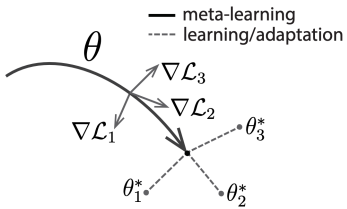


Figure: Meta-learning in MAML [Finn et al., 2017].

Questions to Answer for Yourself / Discuss with Friends

- Activation of what you just learned: Why would you want to preserve the norm of activations and gradients when propagated through the network?
- Transfer: Can you think of some examples of tasks that could be used to perform transfer learning for each other?

Lecture Overview

1. Normalization Layers
2. Transfer Learning
3. General Methodology
4. Parameter Initialization
5. CNN Architectures
6. Detection and Segmentation
7. Summary, Further Reading, References

Foundations of Deep Learning, Winter Term 2021/22

Week 8: Practical Methodology and Architectures

CNN Architectures

Frank Hutter Abhinav Valada

University of Freiburg

LeNet-5 (1998)

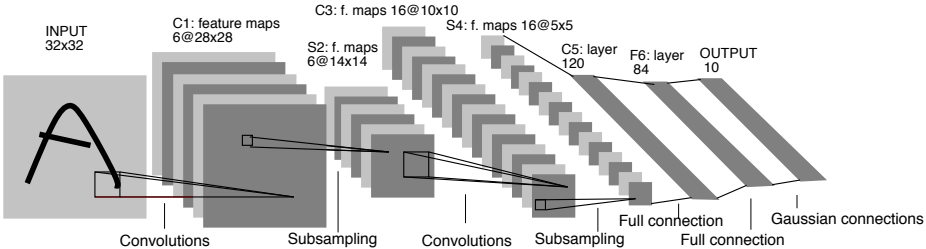


Figure: Architecture of LeNet-5 [LeCun et al., 1998].

- First wave of successful application of CNNs, e.g., reading zip codes, digits, etc.
- Key points:
  - Overall architecture: CONV $\rightarrow$ POOL $\rightarrow$ CONV $\rightarrow$ POOL $\rightarrow$ CONV $\rightarrow$ FC $\rightarrow$ FC
  - Average pooling $\rightsquigarrow$ today, max pooling is used
  - Tanh activation function $\rightsquigarrow$ today, replaced by ReLU and others
- Reminder: Valid convolutions reduce the size (height, width).

The ImageNet Challenge

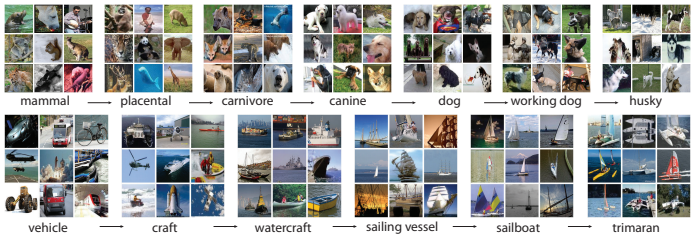


Figure: Examples of the ImageNet dataset [Jia et al., 2009].

- ImageNet Large Scale Visual Recognition Challenge (ILSVRC)
- Annually held from 2010 to 2017.
- Evaluation of algorithms for object detection and image classification at large scale:
  - 1.3M training images
  - 1,000 object classes
- Often used for pre-training.

AlexNet (2012 winner)

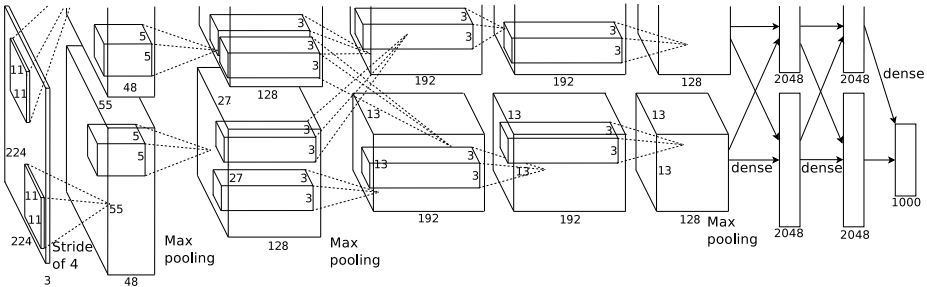


Figure: Architecture of AlexNet [Krizhevsky et al., 2012].

- Popularized CNNs for Computer Vision tasks.
- Key points:
  - Stacking multiple convolutional layers $\rightsquigarrow$ deep networks
  - First use of ReLU activation $\rightsquigarrow$ increase speed and accuracy
  - Normalization layers $\rightsquigarrow$ not commonly used anymore
  - Dropout
  - Heavy data augmentation

ZFNet (2013 winner)

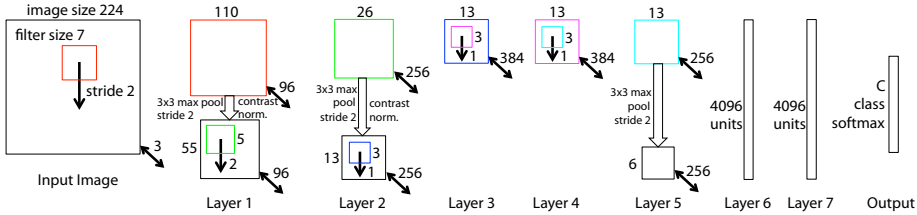


Figure: Architecture of ZFNet [Zeiler and Fergus, 2014].

- Optimize network hyperparameters.
- Key points:
  - Change AlexNet hyperparameters
  - 1st conv layer: decreased filter size and stride
  - Mid conv layers: increased number of channels

VGGNet (2014 runner-up)

- Deep networks with small filters.
- Key points:
  - Stack of three 3×3 filters have the same receptive field as one 7×7 filter.
  - Reduce number of parameters.
    $\rightsquigarrow 3(3^2C^2)$ vs 7^2C^2 (C is channels per layer)
  - Downside: more non-linearities.
  - Downside: many parameters (140M) make inference expensive.
    $\rightsquigarrow$ 1st FC layer has been removed since then without loss in performance.

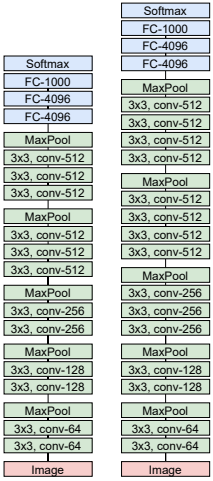


Figure: Architectures of VGGNet-16 and VGGNet-19.

GoogLeNet (2014 winner)

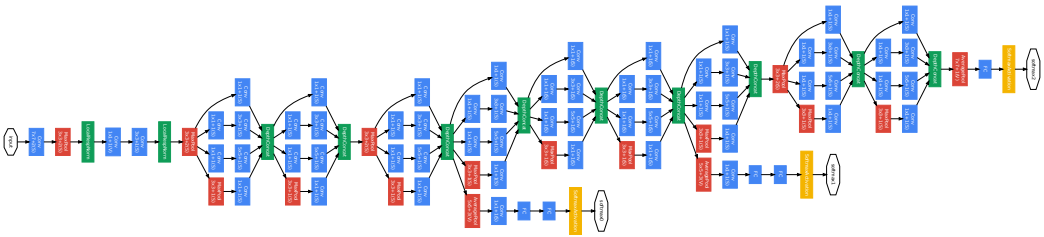


Figure: Architecture of GoogLeNet [Szegedy et al, 2015].

- Focus towards deep networks that are computationally efficient.
- Key points:
  - Inception module $\rightsquigarrow$ reduce number of parameters (AlexNet: 60M, GoogLeNet: 4m)
  - Average pooling instead of FC layers $\rightsquigarrow$ remove parameters without contribution

GoogLeNet – Inception Module

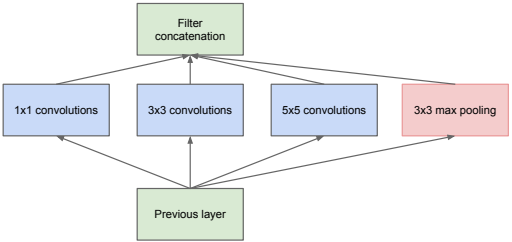


Figure: Naïve version [Szegedy et al, 2015].

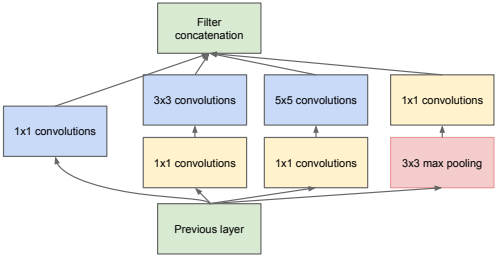


Figure: "Bottleneck" version [Szegedy et al, 2015].

- Design a good local network topology and then stack these modules.
- Key idea: apply parallel filters of different sizes and concatenate the output.
- The naïve version has high computational complexity. $\rightsquigarrow$ very expensive
- The "bottleneck" version reduces the dimensionality via 1×1 convolutions.

ResNet – Motivation

What happens when we continue stacking deeper layers on a "plain" convolutional neural network?

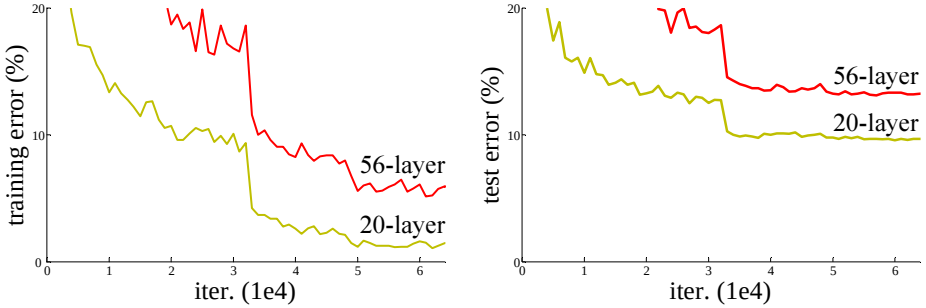


Figure: Error on CIFAR-10 with "plain" networks [He et al., 2016].

The deeper model should be able to perform at least as well as the shallower model.

ResNet – Residual Learning

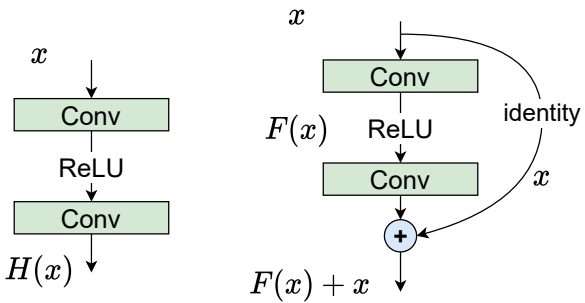


Figure: "Plain" (left) vs residual building block (right).

- Use a small network to fit a residual mapping instead of directly trying to fit a desired underlying mapping. $\rightsquigarrow F(x) = H(x) - x$
- Key idea: make each layer the identity by default.
  - Enable the network to skip layers.

ResNet (2015 winner)

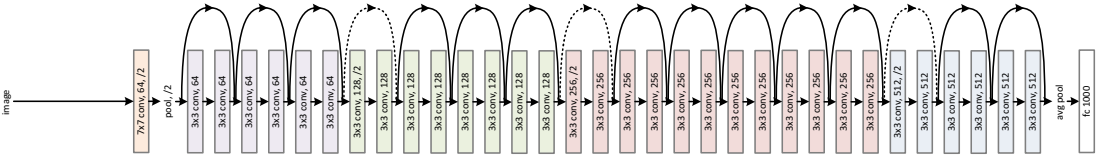


Figure: Architecture of ResNet-34 [He et al., 2016].

- Very deep networks with residual connections.
- Key points:
  - Skip connections (residual building blocks)
  - Batch normalization
  - Only a single FC layer at the end
- Deeper variants, e.g., ResNet-50, use bottleneck blocks.
  $\rightsquigarrow$ Similar idea as Inception module.
- Better than human performance on ImageNet challenge [Russakovsky et al, 2015].

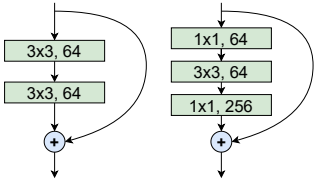


Figure: Normal (left) vs bottleneck block (right).

Variants of ResNet – ResNeXt (2016 runner-up)

Now, focus on efficiency since already superior than human performance.

- From authors of ResNet.
- Increase the width of a residual block by introducing parallel pathways ("cardinality").
- The idea behind the pathways is similar to the Inception module.

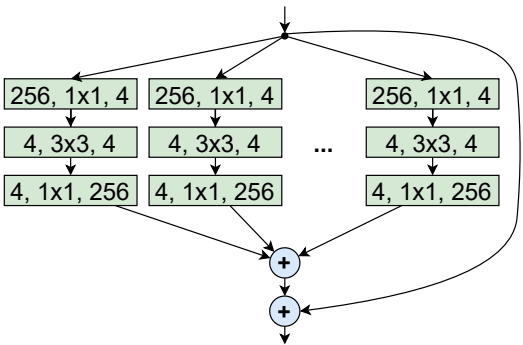


Figure: ResNeXt building block.

Variants of ResNet - DenseNet

Now, focus on efficiency since already superior than human performance.

- Dense block, where each layer is connected to all later layers.
- Advantages:
  - Mitigates vanishing gradient.
  - Encourages feature propagation and re-use.
- Shallow 50-layer can outperform ResNet-152.

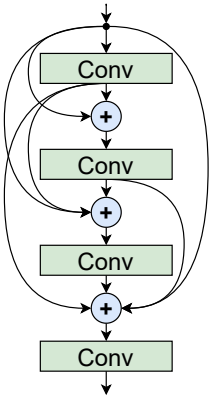


Figure: DenseNet building block.

MobileNets

- Focus towards efficient networks for deployment on mobile devices.
- Replace standard convolutions by two layers (depthwise separable filter):
  - Depthwise convolution
  - Pointwise convolution

  $\rightsquigarrow$ Reduce number of computations and hence increase efficiency.
- Inspired many follow-up works, e.g., MobileNetV2 [Sandler et al., 2018] and ShuffleNet [Zhang et al., 2018].

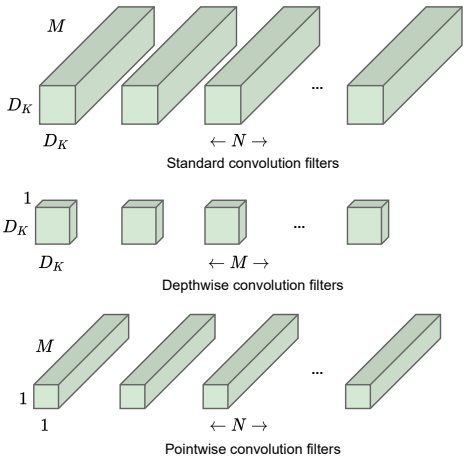


Figure: Depthwise separable filter consisting of depthwise convolutions and pointwise convolutions.

EfficientNet – Structured Model Scaling

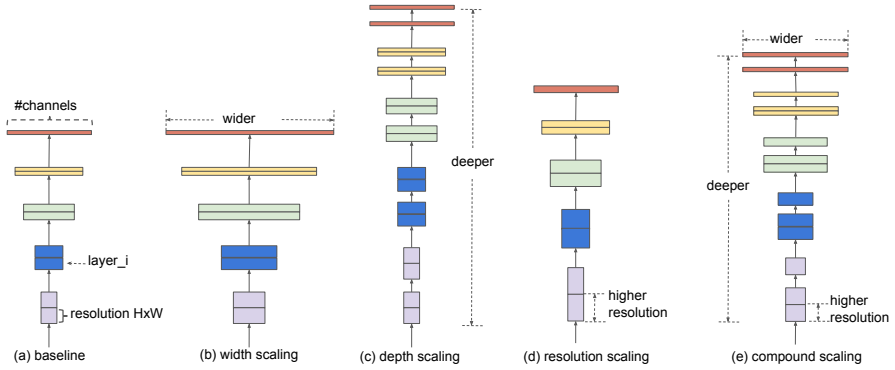


Figure: Various model scaling methods [Tan et al., 2019].

- Commonly, models are developed at a fixed resource cost and then scaled up.
- Conventionally, network dimensions are arbitrarily scaled, e.g., width, depth, etc.
- Compound model scaling balances scaling of dimensions against available resources.

EfficientNet

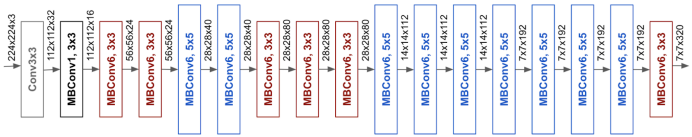


Figure: EfficientNet-B0 baseline architecture.

- Effectiveness of method relies on baseline network.
  $\rightsquigarrow$ EfficientNet allows for simple and clean scaling.
- Key points:
  - Designed via neural architecture search.
  - Using inverted bottleneck convolutions (MBConv), similar to MobileNetV2 [Sandler et al., 2018].

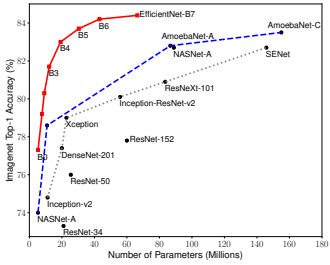
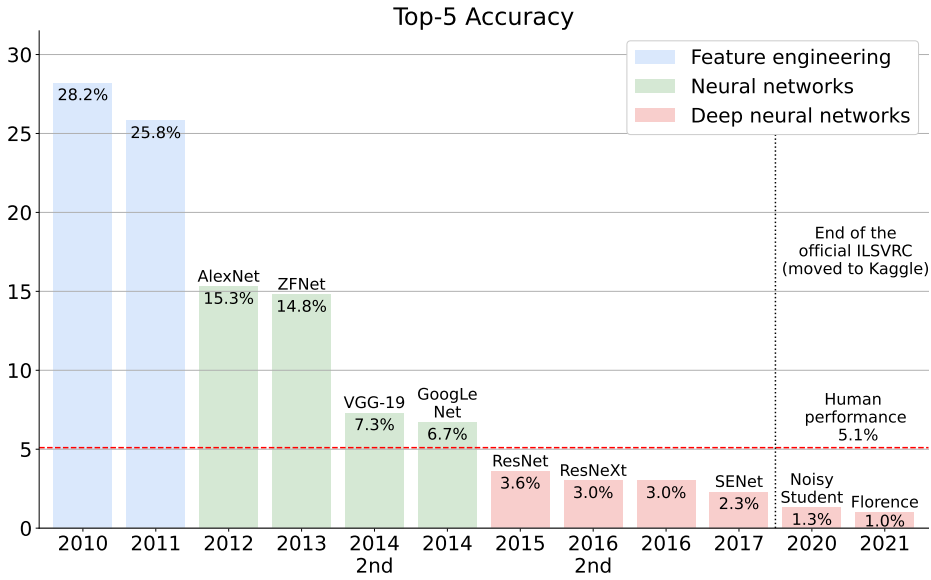


Figure: Model size vs. accuracy comparison [Tan et al., 2019].

ImageNet Challenge – Overview Results



Questions to Answer for Yourself / Discuss with Friends

- Application of what you just learned: What is the best strategy to take while designing CNNs?
- Repetition: What are some methods to keep the number of parameters and computational burden low?
- Repetition: What are some good practices that we can follow to increase the performance of the model in terms of only the metrics?

Lecture Overview

1. Normalization Layers
2. Transfer Learning
3. General Methodology
4. Parameter Initialization
5. CNN Architectures
6. Detection and Segmentation
7. Summary, Further Reading, References

Foundations of Deep Learning, Winter Term 2021/22

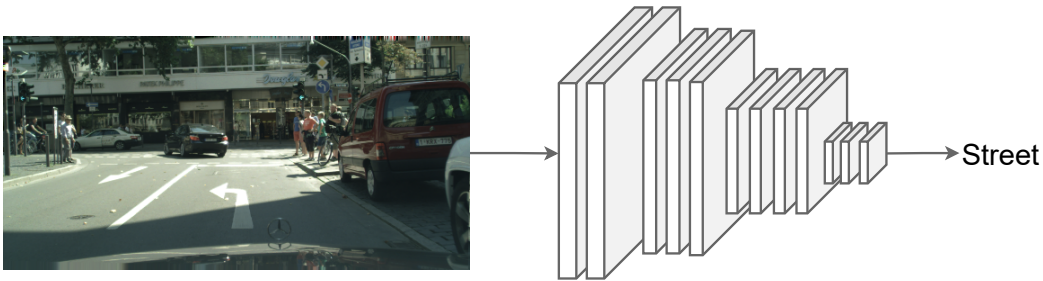
Week 8: Practical Methodology and Architectures

Detection and Segmentation

Frank Hutter Abhinav Valada

University of Freiburg

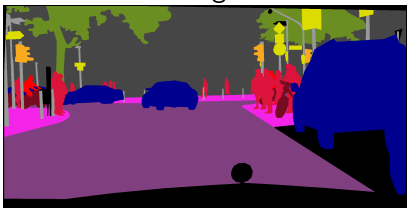
Image Classification



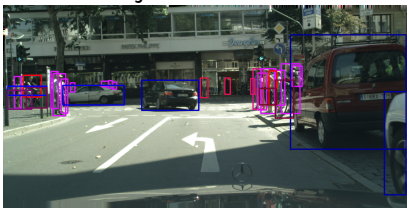
- Assign a single label to the entire image.
- A commonly used benchmark is the ImageNet dataset.

Image Recognition Tasks

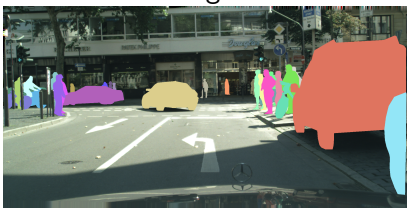
Semantic Segmentation



Object Detection



Instance Segmentation

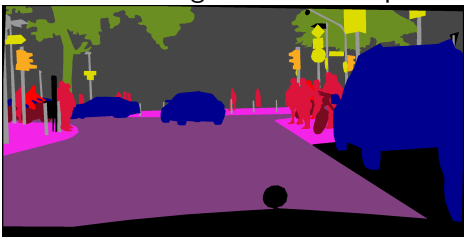


Panoptic Segmentation

Semantic Segmentation

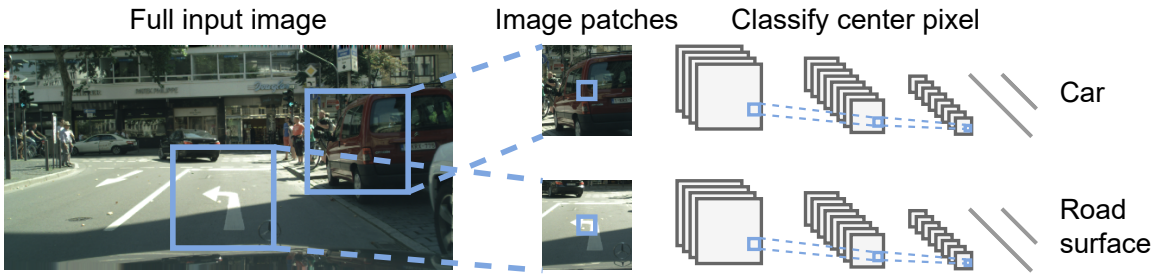
Input Image

Semantic Segmentation Output



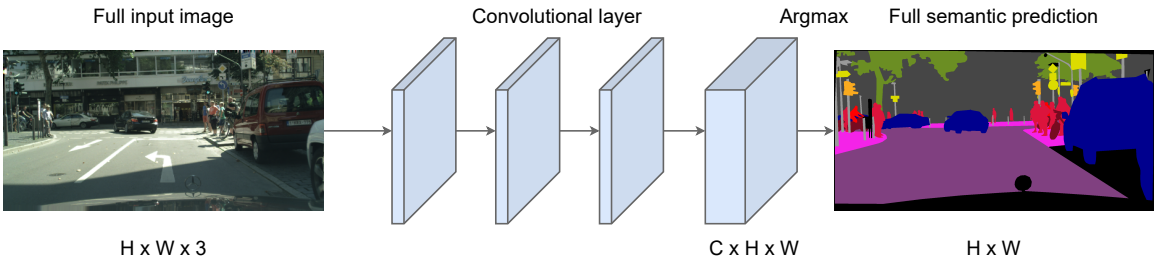
- Assign a class label to each pixel in the image.
- Do not differentiate between different instances of the same class.
- Consider the full scene context, i.e., object classes, location, and shape of all scene elements including the background.

Semantic Segmentation – Sliding Window Approach



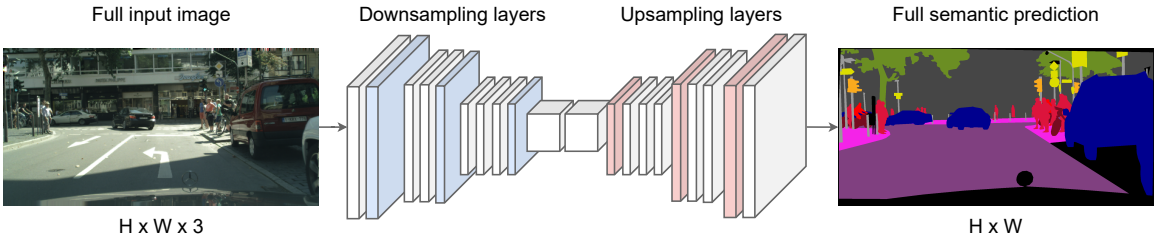
Problem: Very inefficient since overlapping patches do not reuse shared features.

Semantic Segmentation – Fully Convolutional Approach



- Design a convolutional network to predict classes for all pixels simultaneously.
- Problem: Convolutions at the original image resolution will be very expensive.

Semantic Segmentation – Fully Convolutional Approach (cont.)

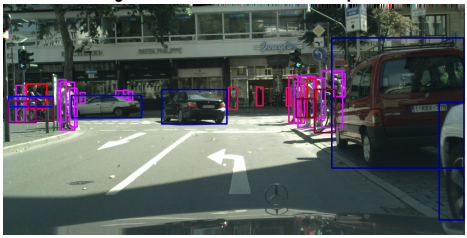


- Design a convolutional network with down- and upsampling layers.
- Downsampling: pooling or dilated convolutions.
- Upsampling: unpooling or transposed convolutions.

Object Detection

Input Image

Object Detection Output



- One of the most fundamental and extensively researched topics in machine vision.
- Requires both identification and localization of objects, e.g., person, vehicles, etc.
- Generate bounding boxes around the given specific target class.

Object Detection – Two-Stage vs. Single-Stage Detectors

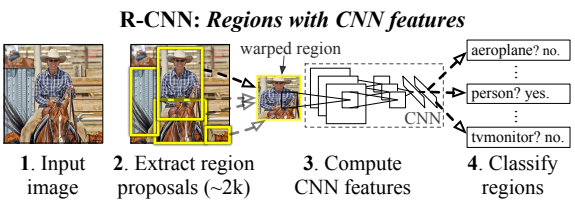


Figure: Two-stage detector, e.g., R-CNN [Ross et al., 2014].

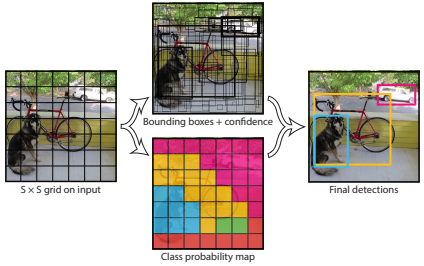


Figure: Single-stage detector, e.g., YOLO [Redmon et al., 2016].

- Two subtasks: object classification and box regression.
- Employ a multi-task loss.
- Two-stage detectors first perform box regression and then classification.
- Single-stage detectors use one CNN for box regression and classification.

Object Detection – Two-Stage Detectors: R-CNN to Faster R-CNN

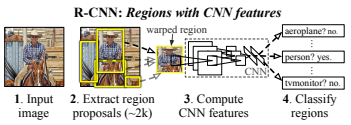


Figure: R-CNN [Ross et al., 2014].

- Extract region proposals from input image.
  No learning!
- CNN-based features for individual boxes.
- SVM for box offset values and object presence.

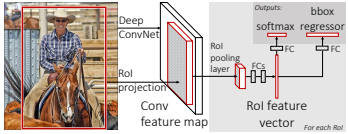


Figure: Fast R-CNN [Girshick, 2015].

- Feed image to CNN.
- Extract region proposals from global feature map.
  Still the bottleneck!
- FCN for regression and classification.

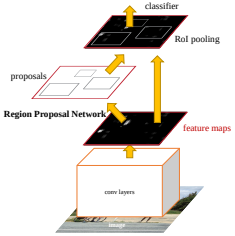


Figure: Faster R-CNN [Shaoqing et al., 2016].

- Learn region proposals with a separate network.

Object Detection – Single-Stage Detectors

Input Image

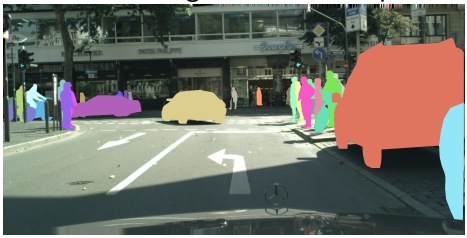
Grid Overlay

- For each grid cell:
  - For each base box, regress a final box with five values: (dx, dy, dh, dw, confidence).
  - Predict scores for each class including the background.
  - Similar to Region Proposal Network but specific for each class.
- Popular examples are YOLO [Redmon et al., 2016], SSD [Liu et al., 2016], and RetinaNet [Lin et al., 2017].

Instance Segmentation

Input Image

Instance Segmentation Output



- Detect and delineate distinct objects of interest in the image.
- Assign different labels for separate instances of the same class.
- Generate pixel-wise annotations.

Instance Segmentation – Mask R-CNN

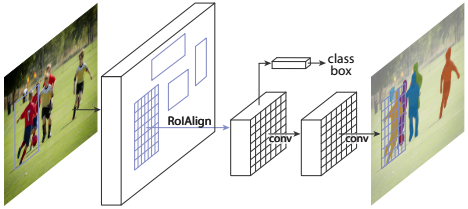


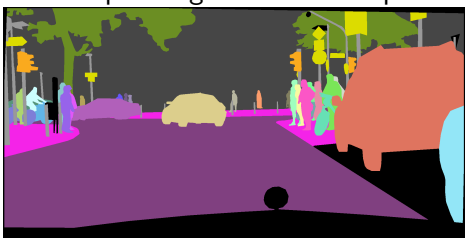
Figure: The Mask R-CNN framework [He et al., 2017].

- When proposed, Mask R-CNN (and its variants) set a long-standing state-of-the-art.
- *Detect-then-segment* approach:
  1) Extract bounding boxes via object detection.
  2) Perform binary segmentation inside each box (object vs. background).
- Builds on top of Faster R-CNN. Problem: Inherits its drawbacks:
  - Slow execution preventing real-time usage.
  - Masks have a fixed resolution.
- Single-stage methods (anchor-free) mitigate these issues, e.g., SOLOv2 [Wang et al., 2020].

Panoptic Segmentation

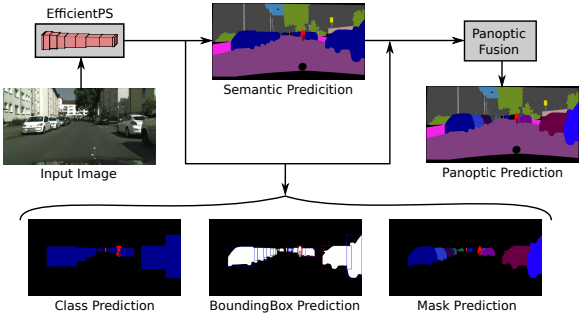
Input Image

Panoptic Segmentation Output



- Combination of semantic and instance segmentation.
- Differ between *stuff* (sky, building, etc.) and *thing* classes (cars, pedestrians, etc.).
- *Stuff* classes receive class labels. *Thing* classes receive both class and instance labels.

Panoptic Segmentation (cont.)



- Most approaches rely on solving multiple sub-tasks.
- *Top-down* methods tackle these in parallel and resolve contradicting results, e.g., overlapping masks, in a fusion module. $\rightsquigarrow$ superior performance
- *Bottom-up* methods use a sequential approach, e.g., semantic segmentation followed by a grouping step to generate instance masks. $\rightsquigarrow$ faster inference speed

[Image source: Mohan and Valada, 2021]

Questions to Answer for Yourself / Discuss with Friends

- Repetition: What is the most naïve approach to perform semantic segmentation with CNNs that you should not do?
- Application of what you just learned: What could be the advantage of performing panoptic segmentation, instead of just doing regular semantic segmentation or instance segmentation?

Lecture Overview

1. Normalization Layers
2. Transfer Learning
3. General Methodology
4. Parameter Initialization
5. CNN Architectures
6. Detection and Segmentation
7. Summary, Further Reading, References

Foundations of Deep Learning, Winter Term 2021/22

Week 8: Practical Methodology and Architectures

Summary, Further Reading, References

Frank Hutter Abhinav Valada

University of Freiburg

Summary by Learning Goals

Having heard this lecture, you can now ...

- describe the different normalization techniques and explain which to use when.
- describe the principle behind transfer learning and explain how you would perform it.
- explain the practical design considerations and debugging strategies.
- describe the reasons for the importance of parameter initialization.
- explain specific practical parameter initialization strategies.
- describe basic architectures of segmentation and detection networks.
- explain how the different image recognition tasks built on top of each other.
- explain the main innovation behind milestone CNN architectures for object classification.

Further Reading

- Resources for this lecture: Deep Learning Book [Goodfellow et al., 2016, chapters 8 & 11]

References

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L. (2009)
ImageNet: A large-scale hierarchical image database
IEEE conference on computer vision and pattern recognition, 248–255
http://vision.stanford.edu/documents/ImageNet_CVPR2009.pdf

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K. (2018)
BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 4171–4186
https://arxiv.org/pdf/1810.04805.pdf

Finn, C., Abbeel, P., Levine, S. (2017)
Model-agnostic meta-learning for fast adaptation of deep networks
International Conference on Machine Learning, 1126–1135
https://arxiv.org/pdf/1703.03400.pdf

Girshick, R. (2015)
Fast R-CNN
Proceedings of the IEEE international conference on computer vision, 1440–1448
https://arxiv.org/pdf/1504.08083.pdf

Girshick, R., Donahue, J., Darrell, T., Malik, J. (2014)
Rich feature hierarchies for accurate object detection and semantic segmentation
Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 580–587
https://arxiv.org/pdf/1311.2524.pdf

Goodfellow, I., Bengio, Y., Courville, A. (2016)
Deep Learning
MIT Press
https://www.deeplearningbook.org

Goodfellow, I., Bulatov, Y., Ibarz, J., Arnoud, S., Shet, V. (2013)
Multi-digit number recognition from street view imagery using deep convolutional neural networks
International Conference on Learning Representations
https://arxiv.org/pdf/1312.6082.pdf

He, K., Gkioxari, G., Dollár, P., Girshick, R. (2017)
Mask R-CNN
Proceedings of the IEEE international conference on computer vision, 2961–2969
https://arxiv.org/pdf/1703.06870.pdf

He, K., Zhang, X., Ren, S., Sun, J. (2016)
Deep residual learning for image recognition
Proceedings of the IEEE conference on computer vision and pattern recognition, 770–778
https://arxiv.org/pdf/1512.03385.pdf

Krizhevsky, A., Sutskever, I., Hinton, G. (2012)
ImageNet Classification with Deep Convolutional Neural Networks
Advances in neural information processing systems 25, 1106–1114
https://dl.acm.org/doi/pdf/10.1145/3065386

References

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P. (1998)
Gradient-Based Learning Applied to Document Recognition
Proceedings of the IEEE 86, 2278–2324
https://www.researchgate.net/publication/2985446_Gradient-Based_Learning_Applied_to_Document_Recognition

Lin, T.-Y., Goyal, P., Girshick, R., He, K., Dollár, P. (2017)
Focal loss for dense object detection
Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2980–2988
https://arxiv.org/pdf/1708.02002.pdf

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016)
SSD: Single shot multibox detector
European conference on computer vision, 21–37
https://arxiv.org/pdf/1512.02325.pdf

Mohan, R., Valada, A., (2021)
EfficientPS: Efficient Panoptic Segmentation
International Journal of Computer Vision 129(5), 1551–1579
https://arxiv.org/pdf/2004.02307.pdf

Noroozi, M., Vinjimoor, A., Favaro, P., Pirsiavash, H. (2018)
Boosting self-supervised learning via knowledge transfer
Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition , 9359–9367
https://arxiv.org/pdf/1805.00385.pdf

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L
Imagenet large scale visual recognition challenge
International journal of computer vision, 115(3), 211–252
https://arxiv.org/pdf/1409.0575.pdf

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2015)
Going deeper with convolutions
Proceedings of the IEEE conference on computer vision and pattern recognition, 1–9
https://arxiv.org/pdf/1409.4842.pdf

Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016)
You Only Look Once: Unified, Real-Time Object Detection
Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 779–788
https://arxiv.org/pdf/1506.02640.pdf

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.-C. (2018)
Mobilenetv2: Inverted residuals and linear bottlenecks
Proceedings of the IEEE conference on computer vision and pattern recognition, 4510–4520
https://arxiv.org/pdf/1801.04381.pdf

References

Ren, S., He, K., Girshick, R., Sun, J. (2016)
Faster R-CNN: towards real-time object detection with region proposal networks

IEEE transactions on pattern analysis and machine intelligence (TPAMI), 1137-1149
https://arxiv.org/pdf/1506.01497.pdf

Tan, M., Le, Q. (2019)
EfficientNet: Rethinking model scaling for convolutional neural networks
International Conference on Machine Learning, 6105–6114
https://arxiv.org/pdf/1905.11946.pdf

Wang, X., Zhang, R., Kong, T., Li, L., Shen, C. (2020)
SOLOv2: Dynamic and Fast Instance Segmentation
Advances in Neural Information Processing Systems 33, 17721–17732
https://arxiv.org/pdf/2003.10152.pdf

Zeiler, M., Fergus, R. (2014)
Visualizing and understanding convolutional networks
European conference on computer vision, 818–833
https://arxiv.org/pdf/1311.2901.pdf

Zhang, X., Zhou, X., Lin, M., Sun, J. (2018)
ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices
Proceedings of the IEEE conference on computer vision and pattern recognition, 6848–6856
https://arxiv.org/pdf/1707.01083.pdf