Weakly-Supervised Segmentation and Unsupervised Modeling of Natural Images

by

Alexander Kolesnikov

A thesis presented to the Graduate School of the Institute of Science and Technology Austria in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Klosterneuburg, Austria.

April, 2018.

Abstract

Modern computer vision systems heavily rely on statistical machine learning models, which typically require large amounts of labeled data to be learned reliably. Moreover, very recently computer vision research widely adopted techniques for representation learning, which further increase the demand for labeled data. However, for many important practical problems there is relatively small amount of labeled data available, so it is problematic to leverage full potential of the representation learning methods. One way to overcome this obstacle is to invest substantial resources into producing large labelled datasets. Unfortunately, this can be prohibitively expensive in practice.

In this thesis we focus on the alternative way of tackling the aforementioned issue. We concentrate on methods, which make use of weakly-labeled or even unlabeled data. Specifically, the first half of the thesis is dedicated to the semantic image segmentation task. We develop a technique, which achieves competitive segmentation performance and only requires annotations in a form of global image-level labels instead of dense segmentation masks. Subsequently, we present a new methodology, which further improves segmentation performance by leveraging tiny additional feedback from a human annotator. By using our methods practitioners can greatly reduce the amount of data annotation effort, which is required to learn modern image segmentation models.

In the second half of the thesis we focus on methods for learning from unlabeled visual data. We study a family of autoregressive models for modeling structure of natural images and discuss potential applications of these models. Moreover, we conduct in-depth study of one of these applications, where we develop the state-of-the-art model for the probabilistic image colorization task.

Acknowledgments

First of all, I want to thank my supervisor, Christoph. I could not have wished for a better supervisor. Ironically, before applying to Christoph's group I had read his group web-page and was intrigued by the breadth and quality of his research, but for some reason assumed that Christoph is likely to be a tough person to work with. Luckily, I knew one of his students, who told me that my assumption is wrong. And it was, indeed, as wrong as it gets. I had absolutely positive experience of being a Christoph's student. Thank you, Christoph, for granting me complete freedom in selecting research topics to work on and for always being available for long detailed discussions regarding my current progress and research obstacles I encountered. Moreover, thank you for introducing me to many bright researches, for gathering a group of extremely talented people, who have taught me a lot, and for maintaining an excellent atmosphere in the group.

I also would like to express my gratitude to my committee members, Vittorio and Vladimir. Thank you for providing me with valuable feedback during my qualification exam, for taking time to share your expertise during the course of my PhD and for your thoughtful comments regarding the text of my thesis. Also, I want to additionally thank Vittorio for hosting and supervising me as an intern in his Google Research team. It was a great experience.

Perhaps, during my PhD I've spent uncountable number of hours with my friends and former office mates, Alex Z, Harald and Michal. I have very vivid memories of our joint activities, such as participating in programming contests, playing football and solving mathematical riddles. Thank you, guys! I also want to acknowledge former and current group members, Georg, Csaba, Emelie, Asya, Mary, Amélie, Nikola, Tomas, Neel, Viktoriia, Saeid, Sylvestre, Nathaniel, Dmitriy, Eela, Chris, Vladislav, Mayu, Sameh for the all fun time we have spent together, and in particular, for our table soccer games.

I would like thank to Amélie, with whom I closely collaborated while working on some results from this thesis. Amélie, I was deeply impressed by your skills in doing research, coding, debugging, writing and I have learned a lot from you.

I am endlessly grateful to my parents for their support of my decision to pursue a PhD degree and to relocate to a foreign country. Finally, I would like thank my wife, Lera, with whom

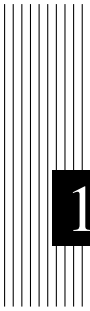
I have shared the most memorable moments and who helped me to withstand the pressure of being a PhD student.

This thesis was partially funded by the European Research Council under the European Unions Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no 308036. I also gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used for this research.

Contents

List of Tables

List of Figures



1. Introduction and Background

Main contributions of this thesis lie in the field of computer vision. Computer vision is a very broad research field, which concentrates on methods for acquiring, processing and understanding visual information. In the thesis we mostly focus on the latter category of methods, i.e. on automatic visual systems that reason about high-level semantic concepts in visual data. These systems have numerous important real life applications, such as automatic inspection at manufactures, event detection (e.g. forest fires), autonomous navigation for robots or self-driving cars, etc.

Despite great practical importance of automatic vision models, developing them is still a challenging scientific and engineering problem. The main challenge arises from the fact that raw visual data is extremely high-dimensional and has very complex structure. For instance, a typical digital image is represented in a computer as an array of millions of numbers. Individually, these numbers do not carry any high-level information about the image. However, as a result of complex high-order interactions, jointly they represent abstract semantic concepts.

As a consequence, contemporary approaches to computer vision heavily rely on statistical machine learning and large-scale optimization techniques. Many classical vision algorithms employ a two-step model, where visual data is first transformed to a more compact and abstract representation and then standard machine learning techniques are applied to learn high-level vision concepts from this representation. These two-step models require moderate amount of training data and they used to form a dominant approach for solving many standard vision task, such as image classification, object detection, etc. However, the performance of these models is relatively poor and often falls far behind performance of human visual systems.

Very recently computer vision research made a tremendous progress by adopting end-to-end trainable models, which jointly learn suitable representation of visual data together with models for reasoning about high-level information in the data. The ability to train such models emerged as a result of multiple important developments. One of them is progress in understanding how to train deep neural networks that have enough capacity to learn useful visual representations. Other developments include growth of the amount of available training data as well as emergence of more powerful hardware, such as largely parallel GPU accelerators.

However, this progress comes at cost of highly increased data requirements. Learning useful representation from raw data requires massive amounts of supervised data. For instance, reliable image classification systems are usually trained from millions of labeled images. At the moment there is a large body of computer vision tasks, for which sufficient amount of training data is not available and, as a result, these tasks can not fully benefit from rapid development of representation learning techniques.

Thus, an important research question is whether data requirements of modern computer vision techniques can be reduced. This question is very broad, so in this thesis we focus on two particular topics. First, we explore a semantic image segmentation task, which is notorious for having expensive to annotate training data. Second, we explore the direction of learning from unsupervised image data. In a sequel of this chapter we introduce these topics in more details and provide a general overview of the thesis.

This chapter provides technical background, which is important for understanding material in the subsequent chapters. Specifically, we review basic topics in machine learning and artificial neural networks. We recommend to consult [45] for more comprehensive overview on these topics. We also present background on semantic image segmentation and unsupervised image modeling.

1.1 Machine Learning Basics

Machine learning is a scientific field which focuses on methods for automatic learning from data. Typically, such systems learn to make predictions about unobserved quantities of interest. More formally, machine learning techniques often aim at estimating an unknown (and possibly probabilistic) function $g(x) : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ is a space of inputs and $\mathcal{Y}$ is a space of outputs. We assume that input-output pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ are distributed according to a data generating distribution $\mathcal{D}$. Then, a learning task is formalized as search for a hypothesis (function) f^* in a fixed set of hypotheses (functions) $\mathcal{F}$, which minimizes the expected loss (penalty for doing wrong predictions):

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \mathcal{L}(f(x), y), \tag{1.1}$$

where $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a loss function. The above expectation is also called *expected risk* in the machine learning literature.

A typical example would be a semantic image classification task. In this task $\mathcal{X}$ is a set of all natural images, $\mathcal{Y}$ is a predefined set of image categories (i.e. dog, cat, person, boat, etc.). A data generating distribution $\mathcal{D}$ is uniform over all image-category pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ and a set of hypotheses is an arbitrary fixed family of parametric functions. A natural choice for a loss function, $\mathcal{L}(f(x), y)$, is the *0-1 loss function*, which is equal to 0 if $f(x) = y$ (i.e. if prediction is

correct) and to 1 otherwise. If a certain $f \in \mathcal{F}$ achieves sufficiently small value of the expected risk in (1.1), then it is guaranteed that this function has high probability of predicting correct label y for a randomly chosen input image x.

Unfortunately, the optimization problem (1.1) can not be solved directly, as the true data generating distribution $\mathcal{D}$ is unknown. Nevertheless, it is possible to derive an approximate solution based on example data sampled from $\mathcal{D}$. In the following sections we will review standard learning approaches for different types of available data.

1.1.1 Supervised learning

We now discuss supervised learning. It is the most studied and widely used branch of machine learning, which provides theoretical basis for understanding other machine learning branches.

In supervised learning a dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ of n independent samples from $\mathcal{D}$ is available to the learner. The learner can utilize this dataset to compute an unbiased estimator of the expected risk from (1.1):

$$\mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \mathcal{L}(f(x), y) \approx \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}(f(x_i), y_i). \tag{1.2}$$

Consequently, the learner can use this estimator, which is also known as *empirical risk*, in order to approximately solve the learning task (1.1) and obtain an approximate solution f^*.

However, there are two potential obstacles when minimizing the empirical loss in (1.2). First, it can be computationally intractable to minimize this objective for the desired loss function $\mathcal{L}$. For instance, consider the 0-1 loss function. This loss function is constant almost everywhere and is notoriously hard to optimize, as local optimization techniques are not applicable. Thus, in practice a smooth approximation $\widehat{\mathcal{L}}$ to the original loss function is often used. We will present an example of such approximation shortly.

Second, it is not generally guaranteed that a solution of (1.2) is a good approximation of the target function g. There are two key reasons for this. On the one hand, the family of hypotheses may not contain any function which is a good approximator of the target function. This effect is called *underfitting*. On the other hand, if the family of hypotheses is too expressive, then there could exist many functions which achieve low value of the empirical risk for the available dataset D, but fail to achieve low value of the expected risk (1.1). Thus, it is likely that the learner selects one of these functions. This effect is known as *overfitting*. In practice, the learner should carefully select the hypothesis set in a way that optimally balances errors from *underfitting* and *overfitting*.

Moreover, it is often beneficial to introduce a model complexity penalty term, $\mathcal{R}(f)$, to the learning objective. This penalty assigns high cost to complex models and serves as way to control a trade-off between underfitting and overfitting errors.

Taking into account all the considerations above, supervised learning is often formulated as the following optimization problem (known as regularized empirical risk minimization):

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \widehat{\mathcal{L}}(f(x_i), y_i) + \mathcal{R}(f). \tag{1.3}$$

Note, that under some relatively mild assumptions it is possible to derive a theoretical upper bound for the expected risk of a function, which is learned by solving (1.3). This bound can be used for principled model selection, i.e. for selecting a hypotheses set, regularization and loss functions in a principled way with guaranteed upper bound on the expected risk. However, in practice this guarantee is often too pessimistic and leads to suboptimal choices. In the next section we discuss an alternative way for model selection.

Model selection and evaluation.

A common practical approach for selecting a hypothesis set $\mathcal{F}$, a loss function approximation $\widehat{\mathcal{L}}$ and regularization function $\mathcal{R}(f)$ relies on splitting available data into two disjoint sets, $D = D_{\text{train}} \bigcup D_{\text{val}}$, which are called training and validation datasets respectively. Given this split, the learner can learn multiple prediction functions f^* using D_{train} for different choices of loss functions, hypothesis sets and regularization functions. Then, for any learned function f^* the learner can use the validation dataset to obtain an unbiased estimator of the expected risk:

$$\frac{1}{|D_{\text{val}}|} \sum_{(x,y) \in D_{\text{val}}} \mathcal{L}(f(x), y) \tag{1.4}$$

Based on this estimator the learner can select the best prediction function f^*, which yields the lowest estimate of the error.

Note, that we can not reuse the validation dataset for reporting final performance of the selected function f^*, as the choice of the this function depends on the validation set. As a result, the validation set does not yield an unbiased estimate of the expected risk. Because of this, final performance of the selected model should be reported on a holdout dataset D_{test}, which does not overlap with D.

Optimization.

We now discuss an algorithm for solving the regularized empirical risk minimization problem (1.3). Generally, solving this optimization problem is non-trivial. Depending on the particular hypothesis set and the loss function different optimization techniques are applicable. In this thesis we make two assumptions regarding the learning task. First, we consider only parametric hypotheses families $\mathcal{F} = \{f_\theta : \mathcal{X} \to \mathcal{Y} | \theta \in \mathbb{R}^d\}$, where d is a number of parameters.

Algorithm 1: Gradient descent optimization algorithm

input : Dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$,
parametric family of functions $\mathcal{F} = \{f_\theta(x) : \mathcal{X} \to \mathcal{Y} | \theta \in \mathbb{R}^d\}$,
cost function $\widehat{\mathcal{L}} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, regularization function $\mathcal{R} : \mathcal{F} \to \mathbb{R}$,
step size α, number of iterations M.

Initialize $\theta \in \mathbb{R}^d$ randomly

For $i = 1 \ldots M$

  $g \leftarrow \nabla \left[\frac{1}{n} \sum_{i=1}^{n} \widehat{\mathcal{L}}(f_\theta(x_i), y_i) + \mathcal{R}(f_\theta) \right]$

  $\theta \leftarrow \theta - \alpha g$

output: θ

Thus, the empirical risk minimization problem over $f \in \mathcal{F}$ can be equivalently formulated as minimization over $\theta \in \mathbb{R}^d$.

Given the above assumptions, the learner can use the gradient descent algorithm in order to approximately solve (1.3). The idea of the algorithm is to start from a randomly initialized parameter vector θ and then to sequentially update the parameters in the direction of the steepest descent of the learning objective. See the listing of Alorithm 1 for a full description.

Note, that assuming sufficiently small step size α, differentiability, convexity and Lipschitzness of the learning objective, gradient descent algorithm is guaranteed to converge to the optimal solution. However, in practice, the gradient descent algorithm is applicable to any almost everywhere differentiable function. In this case it still often works well in practice, even though it may converge to a local optimum.

Example: binary classification with logistic regression.

We now present logistic regression: a concrete example of a popular learning setting. Consider a real d-dimensional vector-valued input space $\mathcal{X} = \mathbb{R}^d$ and a discrete binary output space, where $\mathcal{Y} = \{-1, +1\}$. A hypothesis family $\mathcal{F} = \{\text{sign}(\langle \theta, x \rangle) | \theta \in \mathbb{R}^d\}$ is a set of linear functions (followed by the sign function) parametrized by a real-valued vector θ. A loss function is the 0-1 loss function and its smooth approximation is $\widehat{\mathcal{L}}(y, f(x)) = \log(1 + e^{-y\langle \theta, x \rangle})^*$, which is called *logistic loss*.

Linear logistic regression uses l_2-norm of the weight vector as a regularization term for penalizing model complexity, i.e. $R(f_\theta) = \lambda ||\theta||_2^2$, where λ is a free non-negative real-valued parameter, which controls the strength of regularization.

* We slightly abuse the notation and assume that $\widehat{\mathcal{L}}$ has direct access to θ through the function f.

Given training data $D_{\text{train}} = \{(x_1, y_1), \dots (x_n, y_n)\}$ and parameter $\lambda \geq 0$, linear logistic regression aims to solve the following optimization problem:

$$\theta^* = \underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{n} \sum_{(x,y) \in D_{\text{train}}}^{n} \log\left(1 + e^{-y\langle \theta, x \rangle}\right) + \lambda ||\theta||_2^2. \tag{1.5}$$

When the optimal θ^* is obtained, the learner can estimate the expected error using D_{val} dataset as

$$\frac{1}{|D_{\text{val}}|} \sum_{(x,y) \in D_{\text{val}}} [\![y = \text{sign}\langle \theta^*, x \rangle]\!], \tag{1.6}$$

where $[\![\cdot]\!]$ returns 1 if the argument is true and 0 otherwise.

Based on this estimator of the expected error the learner can select the optimal regularization constant λ. Typically, the optimal λ is selected from a fixed set of constants, which are log-uniformly distributed. For instance, a reasonable search space for λ can be often defined as $\{2^a | a \in \{-10, -9, \dots, 8, 9, 10\}\}$.

1.1.2 Weakly-supervised Learning

In comparison to supervised learning, weakly-supervised learning is a more general and challenging learning setup in which only partial information about target values y is available. A sample data D is given by (X, T), where $X = \{x_1, x_2, \dots, x_n\}$ are samples from $\mathcal{D}$ and T is some partial information about *unobserved* values, $Y = \{y_1, y_2, \dots, y_n\}$, of the target function g. In a case when $T = Y$ we recover the supervised learning setting or when $T = Y' \subset Y$ we get the semi-supervised learning setting [18]. More generally, T can represent arbitrary aggregate information about the target values, for instance T may contain information about frequencies of appearance of various values in Y, without providing information about any individual $y \in Y$.

Evidently, weakly-supervised learning is an extremely general setting and has very wide variety of special cases and corresponding learning techniques. In our review we focus on one particular approach of formulating and solving weakly-supervised problems. We use this approach to tackle the weakly-supervised image segmentation task in the subsequent chapters of this thesis.

Specifically, we tackle weakly-supervised problems through reduction to supervised learning. We derive this reduction by introducing a *supervision* function $\mathcal{S}$, which relies on the all available information about data in order to make the best guess, $\tilde{Y} = \{\tilde{y}_1, \dots, \tilde{y}_n\}$, of the correct values of the unknown supervision Y. Given the supervision function we formulate the

following constrained objective for weakly-supervised learning:

$$f^* = \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \widehat{\mathcal{L}}\left(f(x_i), \tilde{y}_i\right) + \mathcal{R}(f), \tag{1.7}$$
$$\text{s.t.} \quad \tilde{y}_i = \mathcal{S}(x_i, X, T, f) \quad \forall i \in \{1, \ldots, n\}$$

This optimization problem can be tackled using a simple iterative algorithm, which guesses initial random hypothesis f and then alternates between two steps:

1. For the current fixed hypothesis f update values of $\tilde{Y}$ as $\tilde{y}_i = \mathcal{S}(x_i, X, T, f)$ for all $i \in \{1, \ldots n\}$.

2. For the fixed value of $\tilde{Y}$ use supervised learning techniques to update the current hypothesis f. In particular, the learner can use one step of the gradient descent algorithm.

The supervision function $\mathcal{S}$ should be designed by the learner. In this work we rely on three principles when designing a supervision function. First, $\mathcal{S}$ should make use of the available information T and produce those vectors $\tilde{Y}$, which are consistent with the information available in T. Second, among many possible vectors $\tilde{Y}$, which are consistent with T, the supervision function should prefer one, which will minimize learning objective (1.7) for the current value of f. Finally, the supervision function may make use of prior information in order improve quality of a label guess $\tilde{Y}$.

In some cases it may be problematic to specify a single supervision $\mathcal{S}$ function, which adequately leverages all of the available knowledge about target values Y. Instead, it may be easier and more beneficial to specify multiple complementary supervision functions, which are responsible for incorporating various pieces of available information about the target values Y. In this case the final objective of the weakly-supervised takes form of sum of multiple losses, each of which has corresponding supervision function $\mathcal{S}$.

Example: Multiple Instance Learning (MIL).

In this section we discuss Multiple Instance Learning, an example a weakly-supervised task, which was initially introduced in [37]. We consider a case, where targets are the binary labels (i.e. $\mathcal{Y} = \{-1, 1\}$), and the input space is a real space, i.e. $\mathcal{X} = \mathbb{R}^d$. In the MIL setting a sample data $X = \{x_1, \ldots, x_n\}$ and weak-supervision T can be represented as a collection labeled bags, i.e. $\{(B_1, u_1), (B_2, u_2), \ldots, (B_b, u_b)\}$, where $B_i = \{x_1^i, \ldots, x_{b_i}^i\}$ is a subset of X and $u_i \in \{-1, +1\}$ is a bag label for all $i \in \{1, \ldots, b\}$. Each training example $x \in X$ belongs to one of the bags. If a bag is labeled by -1, then this implies that all instances inside this bag have label -1. However, if a bag has label +1, then this implies that at least one object in this bag has label +1.

As we have discussed, assuming that the supervision function $\mathcal{S}$ is defined, the MIL task can be reduced to a supervised learning problem. In order to setup the supervised learning problem, we use exactly the same choices as in the previous example about logistic regression. In particular, we use a family of linear functions, $\mathcal{F} = \{\text{sign}(\langle\theta, x\rangle)|\theta \in \mathbb{R}^d\}$, as the hypothesis set and the logistic loss for training.

It is left to define the supervision function $\mathcal{S}$, which for any given hypothesis θ is defined by the following rules:

- If x belongs to a bag labeled with -1, then $\mathcal{S}$ always returns a label -1 for this object.
- If x belongs to a bag labeled with +1 and has the highest score $\langle\theta, x\rangle$ among other objects in this bag, then S assigns a label +1 to this object.
- Finally, if x belongs to a bag labeled with +1 and does not have the highest score $\langle\theta, x\rangle$ among other objects in this bag, then $\mathcal{S}$ assigns a label equal to $\text{sign}(\langle\theta, x\rangle)$.

This definition of $\mathcal{S}$ guarantees that the resulting label guesses are always compatible with the available bag-level supervision. At the same time, this supervision function also strives to guess a labeling, which is consistent with the current hypothesis θ.

1.1.3 Unsupervised Learning

Unsupervised learning is an extreme case, where no information about target values y is available, i.e. only $X = \{x_1, x_2, \ldots, x_n\}$ is available to the learner. In this case there is little hope of learning an unknown function $g : \mathcal{X} \to \mathcal{Y}$. However, the sample data X can be used to learn structure of the inputs from $\mathcal{X}$. In order to understand why this is useful consider the following example: archaeologists discover a book, which is written in an unknown ancient language. Ideally, they would like to translate this book to one of the known languages, e.g. English. However, there is no supervised data, which demonstrates how the unknown language can be translated to English. Nevertheless, by using prior knowledge about other languages, linguists can potentially learn a lot about the an unknown language just by studying linguistical structures, which appear in the book, and, hopefully, they can partially understand the contents of the book.

A common approach to unsupervised learning is to formulate an auxiliary learning task, which only needs unsupervised data X and strives to understand data structure in order to minimize a certain learning objective. We illustrate this approach by giving an example in the next section.

Example: Auto-encoding model

Auto-encoding model is based on an elegant idea of an auxiliary task, which strives to learn the structure of unsupervised data by learning how to compress this data. The auxiliary objective of auto-encoding model is to learn an identity function, i.e. the target function g is given by $g(x) = x$. The loss function, e.g. squared Euclidean distance $||x - f(x)||_2^2$, measures the reconstruction error.

Generally, this is a trivial task, as the majority of hypotheses sets, which are used by machine learning practitioners, contain the identity function. However, the hypothesis set of the auto-encoding task is constrained in order to encourage learning a non-trivial identity function. Concretely, in the auto-encoding learning task any hypothesis f can be written as a composition of two functions, $f(x) = \text{dec}(\text{enc}(x))$, where $\text{enc} : \mathcal{X} \to \mathcal{Z}$ and $\text{dec} : \mathcal{Z} \to \mathcal{X}$. Crucially, it is required that the dimensionality of $\mathcal{Z}$ is much smaller than the dimensionality of $\mathcal{X}$. With these constraints f should learn how to compress the input data to a small space $\mathcal{Z}$ and then decompress it back.

Thus, the encoder strives to learn a compact representation of x, which ignores redundant information and compactly encodes information about the input object x. This representation may uncover underlying structure of the input data and it can be later used to solve supervised or weakly-supervised learning problems more efficiently, when a small amount of supervised data becomes available.

1.2 Artificial Neural networks

In this section we review artificial neural networks, a powerful class of parametric models that achieve state-of-the-art performance for many important tasks in computer vision, speech recognition, natural language processing and in other domains.

A design of the artificial neural networks is inspired by ideas of *connectionism* [41], which hypothesizes that complex intelligent systems can be modeled by a large amount of elementary interacting computational units. On a high level, a neural network is a parametric vector-valued function, which consists of a large amount of computational units called *neurons*. Neurons are connected to each other by directed *connections* and each connection has a scalar weight. A valid neural network can not have cycles in its connectivity structure.

Every neuron is associated with a simple rule for computing its value, see Figure 1.1. A value, v, of any neuron is computed from values of all neurons, which have connections pointing to this neuron. Assuming that there are k connected neurons with already known values $\{v_1, v_2, \ldots, v_k\}$ and weights of the corresponding connections are $\{\theta_1, \theta_2, \ldots, \theta_k\}$, then v is

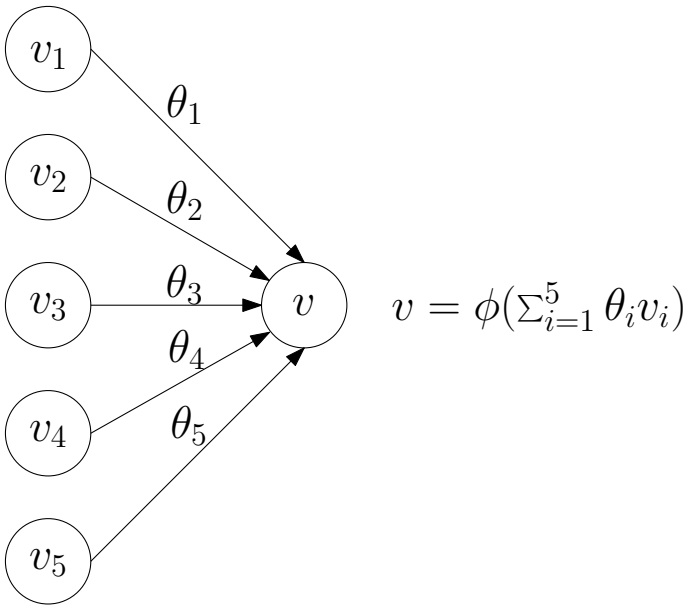


Figure 1.1: A schematic illustration of how the value, v, of a neuron is computed. The circles depict neurons and the arrows depict directed connections.

computed as $\phi(\sum_{i=1}^{k} \theta_i v_i)$, where $\phi : \mathbb{R} \to \mathbb{R}$ is an arbitrary scalar function, which is called an activation function. Neurons, which do not have any incoming connections, are called *input* neurons. Values of these neurons need to be set by a learner and, in practice, these neurons are used to feed in input examples x. The output, y, of the network is constructed as concatenation of an arbitrary fixed subset of neurons, which are called *output* neurons. Given the above description, an artificial neural network can be seen as a parametric function, which implements vector-valued functions $f_\theta : \mathcal{X} \to \mathcal{Y}$.

Neurons in neural networks are often organized in a form of multiple **ordered** *layers*. A *layer* is a collection of neurons. Every neuron should belong to one of the layers. By convention, the order of layers imposes a constraint, which prohibits connections pointing from neurons from layers with a higher rank to neurons in layers with a lower rank. Note, that this constraint is not restrictive, as neurons of any valid neural network with arbitrary connectivity structure can be arranged in layers, such that the constraint is satisfied.

Nevertheless, the ordered layer-wise interpretation is a very useful abstraction. It allows to think about neural networks as of modular objects, which consist of multiple layers stacked on top of each other. By convention, all input neurons go to first layer, which is called the *input layer*, while all output neurons form the output layer, which is also the last layer. All intermediate layers are called hidden layers.

The connectivity structure between layers is a very important characteristic of neural networks and has crucial impact on the network's performance in practical applications. In the following we will review two common classes of neural networks with different connectivity

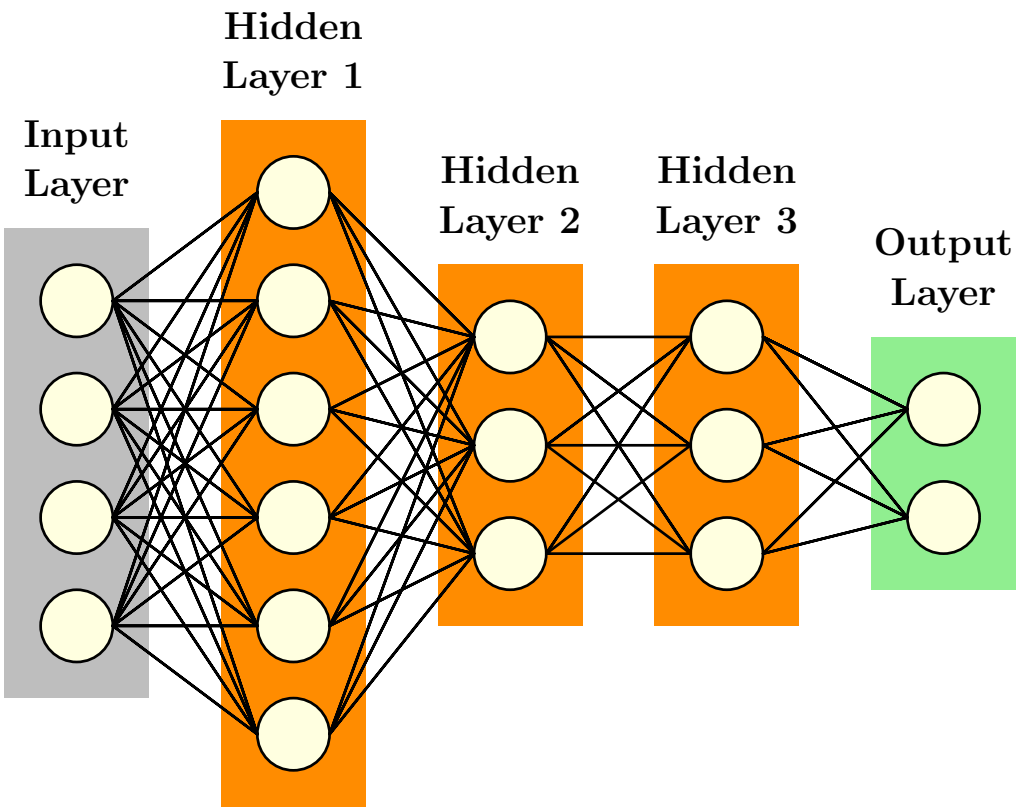


Figure 1.2: An illustration of a fully-connected neural network with three hidden layers.

structures.

1.2.1 Neural networks with fully-connected layers

Neural networks with fully-connected layers form an important class of artificial neural networks, which are widely used in practice and have a solid theoretical foundation. These networks have a simple connectivity structure, which, nevertheless, results in an efficient and expressive parametric family of functions. For historical reasons a network of this type is also called *MultiLayer Perceptron* or *MLP* in the literature on neural networks.

A typical MLP is illustrated in Figure 1.2. By the defenition of MLP, any neuron, except for those in the output layer, is connected to all neurons in the next layer and only to them. This constraint imposes minimalistic assumptions on the connectivity structure and, yet, yields the expressive, efficient and easy-to-analyze family of neural networks. Assuming that all neuron values in a certain layer are stored in a vector, u, then values of the consecutive layer, v, can be computed as a matrix-vector multiplication followed by element-wise application of the activation function:

$$v = \phi(\Theta u), \tag{1.8}$$

where Θ is a matrix of connections' weights. A value of Θ_{ij} is equal to a weight of the connec-

tion between neuron number i in j in the layers u and v respectively.

Thus, a neural network with fully-connected layers can be implemented as a series of matrix-vector multiplications followed by the activation function. As a result, MLP can be efficiently implemented and parallelized by using standard libraries for matrix-vector multiplication.

Note, that it is important that the activation function is smooth (at least almost everywhere) and non-linear. The smoothness property is important for enabling efficient gradient-based parameter learning. The non-linear property is also crucial, because otherwise the resulting MLP collapses into a linear function. A popular choice, which is often used in the neural networks is the so-called ReLU activation function [77], which is given by $\phi(v) = \max(0, v)$.

The simple mathematical description of networks with fully-connected layers allows for thorough theoretical analysis. In particular, under some mild assumptions it is possible to show that MLP with only one hidden layer is a universal functional approximator [27]. This means, that given enough neurons in the hidden layer, MLP can approximate any continuous vector-valued function arbitrary well. It is an important results, which partly sheds light on the success of artificial neural networks.

1.2.2 Convolutional neural networks

One drawback of the networks with fully-connected layers is the number of connections and their weights, which scale quadratically with the number of neurons in layers. Thus, large inputs result in the excessive number of connections and lead to computational inefficiency and poor performance in learning tasks. These issues become especially pronounced for processing visual data, which may have millions of input dimensions.

In this section we present convolutional neural networks. These networks address the aforementioned shortcomings of MLPs by using convolutional and pooling layers instead of the fully-connected layers. Note, that for clearness of presentation we restrict our discussion of convolutional neural networks by the context of computer vision applications. However, convolutional neural networks are also successful in many other domains, such as audio processing or natural language understanding.

Convolutional layer

A convolutional layer can be seen as a special case of the fully-connected layer. Convolutional layers leverage spatial structure of visual data and differ from the standard fully-connected layers by imposing constraints on the weight matrix Θ. These constraints make use of local structure of visual inputs and enforce sparsity of the weight matrix. Convolutional layer also

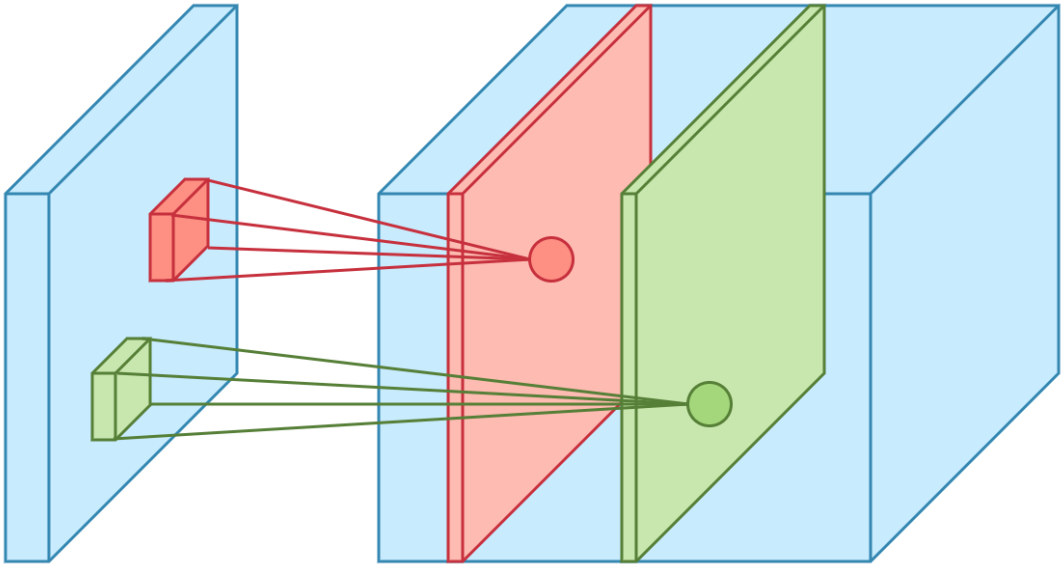


Figure 1.3: An illustration of the convolutional layer. *Image credit: Arden Dertat*.

leverages translation-invariant properties of visual data by sharing weights between different connections. For instance, if, let's say, a cat is present in the image, then it will have the same visual appearance regardless of where exactly it appears in the image.

Figure 1.3 illustrates the structure of a convolutional layer. Input neurons *(on the left part of the figure)* and output neurons *(on the right part of the figure)* are organized in a form of three-dimensional structures. For concreteness we assume that input neurons are represented by a collection of neurons of the shape $H \times W \times K$ and output neurons have shape $H \times W \times K^{'}$. The dimensions of size H and W are spatial and should be interpreted as width and height dimensions. For instance, an RGB-image with resolution 640x480 can be represented as a three-dimensional structure of neurons with $K = 3$ and $H = 640$ and $W = 480$.

A convolutional layer consists of multiple channels of the size $H \times W$. In Figure 1.4 two such channels are illustrated as green and red slices inside the output layer. Each channel is associated with a real-valued weight tensor (also called kernel), of size $w \times h \times K$, where w and h are, typically, small numbers in a range $[3, 7]$. Every neuron within a convolutional channel is connected to its spatial neighborhood of size $w \times h \times K$ in the input layer and the weights of these connections are given by the kernel corresponding to this channel. In total, a complete convolutional layer is defined by $w \times h \times K \times K^{'}$ free parameters.

In order to compare a fully-connected layer with a convolutional layer consider a concrete (and typical for contemporary applications) example of input and output layers with sizes given by $H = W = 224$ and $K = K^{'} = 64$. In this case the number of free weight parameters in the fully-connected layer will be, approximately, 10^{13}. Contrary, assuming that $w = h = 3$, the number of parameters in the convolutional layer is 36864, which is much smaller.

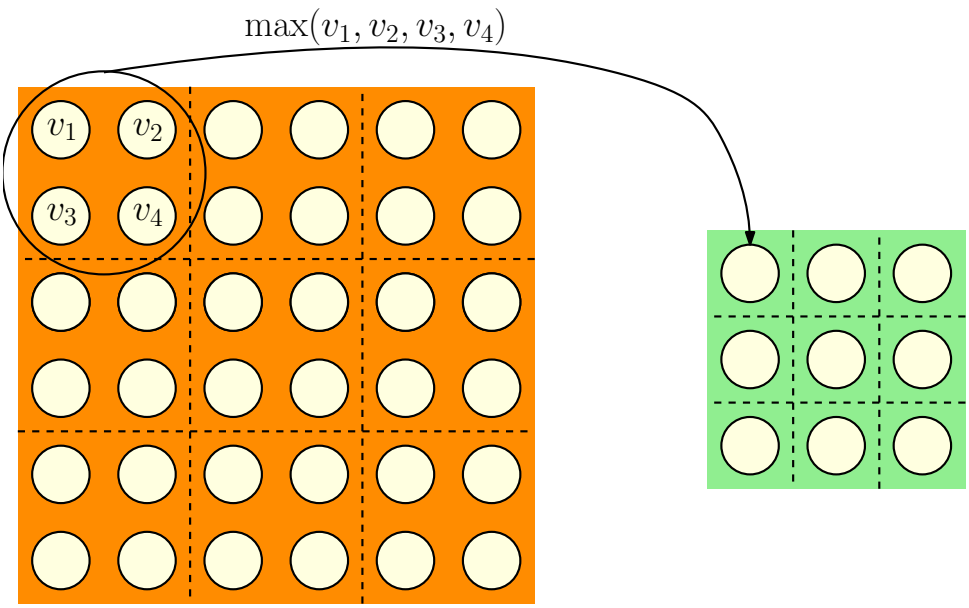


Figure 1.4: An illustration of the max-pooling layer.

Pooling layer.

A pooling layer is commonly used to reduce spatial resolution of layers in convolutional neural networks.

Like the convolutional layer, it operates with input neurons organized in a three-dimensional grid of size $H \times W \times K$. The objective of the pooling layer is to reduce the amount of spatial information in order to even further improve computational efficiency and reduce the number of free parameters. It does so by splitting every input channel of size $H \times W$ into a regular grid of cells of the fixed size, see left part of Figure 1.4 for the illustration of a grid with 2×2 cells. Then, it aggregates all values within every cell into one value. As a consequence, the pooling layer reduces the spatial dimensions of the input layer by a factor of 2.

The popular aggregation rule is a maximum function over all values in a cell. In this case the resulting operation is called max-pooling. Max-pooling layer reduces the spatial size by keeping only high-scoring neurons within small spatial neighborhoods and ignoring other neurons. Max-pooling is often used because it produces outputs that are invariant to a small spatial perturbations of the input data, which is often a desired property of computer vision models.

Another widely used pooling layer is an average pooling layer. It aggregates neurons by computing their average value. The advantage of the average pooling layer compared to the max-pooling layer is that all neurons contribute to the values of the next layer. This property may be beneficial for some applications, such as semantic image segmentation [160].

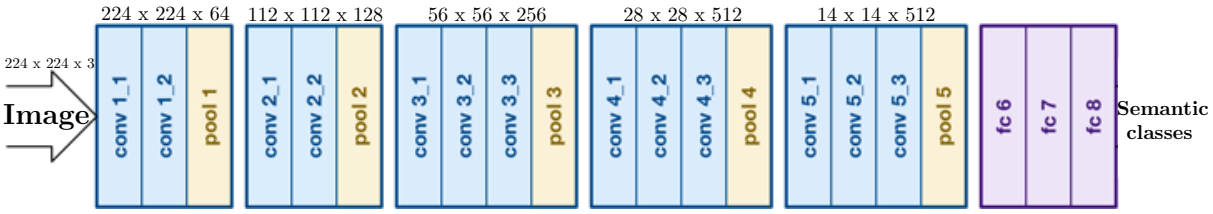


Figure 1.5: A schematic illustration of a modern convolutional architecture with 13 convolutional layers, 5 max-pooling layers and 3 fully-connected layers. The sizes of convolutional layers are indicated on top of them: the first two numbers represent spatial dimensions and the last one indicates the number of channels. This architecture was proposed by [134]. *Image credit: Matthijs Hollemans*.

Example of a modern convolutional architecture

Modern convolutional neural networks typically consist of dozens or hundreds of convolutional layers and, optionally, a few pooling and fully-connected layers. A concrete example of a simple and successful modern convolutional architectures is the *VGG-16* model, which is illustrated in Figure 1.5. This network learns to solve image classification task with 1000 semantic categories. It has 13 convolutional layers (with $w = h = 3$), 5 max-pooling layers and three fully-connected layers. Notably, a large number of layers is necessary to achieve the best classification performance.

The input of this network is an RGB image of size $224 \times 224 \times 3$ and the output is a vector of semantic classes' probabilities. The sizes of the convolutional layers are illustrated in Figure 1.5. The fully-connected layers have 4096, 4096 and 1000 neurons respectively. See the original paper [134] for more details on this architecture.

1.2.3 Parameter learning in neural networks

Optimizing the learning objective (1.3) for models parametrized by contemporary neural networks poses many computational challenges. For instance, only one evaluation of the gradient of a learning objective may take multiple hours, if not days. This is a consequence of a large size of standard computer vision datasets, which often have millions of images, and highly expressive models, which are represented by deep neural networks with potentially hundreds of convolutional layers.

Because of this, contemporary neural networks are often trained with stochastic gradient descent, which, in practice, converges to a good solutions much faster than the standard gradient descent algorithm. The main idea behind stochastic gradient descent is to use noisy, but cheaper-to-compute gradient estimator instead of computing the real gradient. Each update step of stochastic gradient descent uses small random subset of data $D' \subset D$ and computes the

Algorithm 2: Stochastic gradient descent optimization algorithm

input : A dataset $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$,
parametric family of neural networks $\mathcal{F} = \{f_\theta(x) : \mathcal{X} \to \mathcal{Y} | \theta \in \mathbb{R}^d\}$,
cost function $\widehat{\mathcal{L}} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, regularization function $\mathcal{R} : \mathbb{R}^d \to \mathbb{R}$,
batch size m, step size α, number of epochs M.

Initialize $\theta \in \mathbb{R}^d$ randomly
For $i = 1 \ldots M$
    Shuffle D
    Split data in batches of size m: $B_1, B_2, \ldots, B_{\lfloor n/m \rfloor}$
    For $j = 1 \ldots \lfloor n/m \rfloor$
        $g \leftarrow \nabla \left[\frac{1}{|B_j|} \sum_{(x,y) \in B_j} \widehat{\mathcal{L}}(f_\theta(x), y) + \mathcal{R}(f_\theta) \right]$
        $\theta \leftarrow \theta - \alpha g$

output: θ

following estimator of the real gradient:

$$\nabla \left[\frac{1}{n} \sum_{(x,y) \in D} \widehat{\mathcal{L}}(f(x), y) + \mathcal{R}(f) \right] \approx \nabla \left[\frac{1}{|D'|} \sum_{(x,y) \in D'} \widehat{\mathcal{L}}(f(x), y) + \mathcal{R}(f) \right] \tag{1.9}$$

See Algortihm 2 for a full description of stochastic gradient descent. Importantly, this optimization algorithm is not just a heuristic, as it has formal theoretical guarantees regarding convergence [12].

Researches and practitioners often use modifications of stochastic gradient descent, which may result in faster convergence for many practical application. Some popular modifications include adding Nesterov momentum [90] or using adaptive learning rates [66].

Deriving analytic gradient of an artificial neural network is very cumbersome and prone to many errors. Because of this it is recommended use software packages that automatically derive and compute gradients of neural networks. Some prominent examples of these packages are *theano*, *caffe*, *tensorflow* and *pytorch*. These packages rely on the backward-mode automatic differentiation technique, which provides an efficient procedure for computing gradients of a learning objective function with respect to unknown parameters

Note, that in addition to using improved optimization techniques, researchers recently proposed several important architectural modifications for convolutional neural networks. These modifications greatly facilitate faster parameter learning and numerical stability. The two most prominent modification are batch normalization [59] and skip-connections [54].

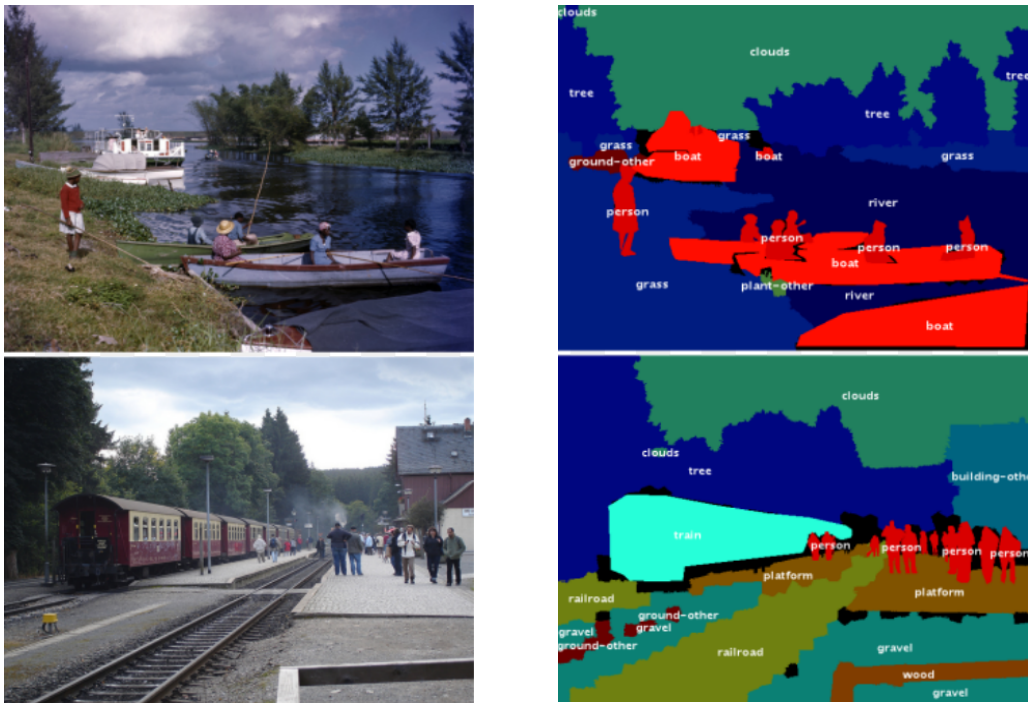


Figure 1.6: Illustration of inputs and outputs of a semantic image segmentation model. Inputs are given by raw RGB images. Outputs are dense segmentation masks, which provide pixel-level decomposition of images into semantic regions, such as *people, sky, train etc. Image credit: COCO-stuff dataset [14].*

1.3 Semantic Image Segmentation

Semantic image segmentation is an important vision task, which is essential for comprehensive scene understanding. Given an input image, a semantic image segmentation model should produce a dense segmentation mask, which partitions the image into semantic regions. Figure 1.6 illustrates typical inputs and outputs of image segmentation models. The set of semantic categories is typically fixed and there are two common types of categories: *things (e.g. cars, cups, horses, etc.)* and *staff (e.g. sky, grass, road, etc.)*. Some formulations of the semantic image segmentation task include a special *background* class, which represents all image pixels that do not belong to the current set of semantic classes.

Image segmentation with structured models. Image segmentation is known to be a very complex task as it requires dense semantic analysis of high-dimensional raw visual data. The first relatively successful framework for tackling image segmentation tasks emerged in the 2000s. This framework relies on supervised machine learning techniques, hand-crafted local image feature descriptors and learning/inference techniques for structured models [95]. Prominent and widely used examples of structured models are Conditional Random Field (or CRFs) [78] and structured support vector machines [140].

Local image feature descriptors are designed to provide concise high-level information about raw visual data. Typically these features describe local distribution of image gradients. The examples of widely used descriptors are SIFT [86], HOG [30] or SURF [6]. These descriptors can be enhanced by concatenation of additional information, such as distribution of local image colors or relative image coordinates.

Structured models allow to model high-order segmentation properties, such as local smoothness of segmentation masks [116], connectivity [94] or convexity [47] of segmented regions and many others. Note, that inference in structured models is computationally intractable in general, so approximate inference techniques are often used.

The aforementioned semantic image segmentation framework can be summarized in four main steps:

1. Employ a hand-crafted image descriptor (e.g. SIFT) to compute dense image feature representation.

2. Formulate a parametrized structured model (e.g. CRF), which models the dependency between local image descriptors and semantic labels as well as higher-order label interactions, such as smoothness.

3. Learn unknown parameters using available supervised data.

4. At the test time for any given input image use the learned probabilistic graphical model (e.g. CRF) to produce the most probable segmentation mask.

A prominent example of an approach, which follows this framework, is [131]. There are numerous improvements of this general approach in the literature. For instance, [48] proposes a technique that dynamically groups neighboring pixels into semantically and geometrically consistent regions, which are then labeled by semantic labels. Another example is [109], which proposes to use global image semantic context in order to rule out semantic labels that do not fit into the overall image context.

Nevertheless, structured prediction models have multiple crucial drawbacks. First, they rely on hand-crafted image descriptors, which often do not carry enough information to discriminate certain semantic categories. Second, inference and parameter learning in structured probabilistic models is often slow or even computationally intractable, thus requiring careful development of specialized approximate learning or inference techniques. Overall, these models are quite slow in practice and their performance falls far behind the performance of human visual systems.

Image segmentation with deep neural networks. Recently, a new paradigm based on deep convolutional neural networks became a dominant choice for solving semantic segmentation

task for natural images. It relies on end-to-end training of a convolutional neural network, which maps input images to segmentation masks. Unlike structured models based on hand-crafted image descriptors, deep neural networks learn image representation automatically using available data.

Authors of [85] demonstrate that modern convolutional deep neural networks significantly outperform structured models. Subsequently, their model was revised and substantially improved. A paper [22] proposes to use dilated convolutions, which drastically increase the size of network's receptive field without increasing the number of free parameters. In [170] authors develop a methodology for fusing deep neural networks and the fully-connected CRF model from [74] in the end-to-end fashion during both training and evaluation phases.

1.3.1 Weakly-supervised semantic image segmentation

What makes semantic image segmentation especially challenging is the cost of producing labeled data in a form of dense segmentation masks. Each image requires at least a few minutes of a trained human annotator to be fully annotated. This makes the task of producing a large image segmentation dataset to be prohibitively expensive in many realistic situations. Thus, a large body of previous research is devoted to weakly-supervised models that can learn from much weaker (and cheaper to produce) forms of annotation. In particular, image-level labels are much cheaper to produce and, thus, this type of weak annotation have attracted a lot of attention in the computer vision research community.

One of the first successful attempts to learn semantic image segmentation model from image-level labels is [149]. It is based on a probabilistic structured model, which combines a signal from image-level labels with the prior assumption on label smoothness. Smoothness is enforced for spatially neighboring superpixels within one image as well as for similarly looking superpixels across images. Moreover, the authors utilize objectness prior [1] to further improve the segmentation quality. A follow-up paper [150], extends this approach by introducing additional hyperparameters for controlling importance of various terms in a structured segmentation model. Importantly, authors also propose a novel criteria for selecting these hyperparameters, which does not require fully-labeled segmentation data to be evaluated.

Emergence of powerful deep convolutional neural networks sparked a rapid progress in the performance of weakly-supervised image segmentation models. In [104] authors derive weakly-supervised segmentation model by combining MIL framework [2] with a deep convolutional neural network for image segmentation. A follow-up work [105] improves the design of a MIL loss function and, additionally, introduces image segmentation priors, which result in substantial performance gains. Another line of work [100, 102] utilizes variants of Expectation Maximization algorithm [31] (or EM-algorithm). The main idea is to iterate between

two steps: expectation step and maximization step. At the expectation step the learner combines image-level labels and segmentation predictions produced by the current approximation of a segmentation model in order to produce surrogate ground-truth segmentation masks. At the maximization step the current segmentation model is updated in order to better match the surrogate ground-truth segmentation masks. The proposed EM-algorithm based models mostly differ in the exact way of how the expectation step is performed.

Overall, recently proposed weakly-supervised segmentation models deliver roughly 60% of the performance of fully-supervised analogues. In this thesis we aim to make this performance gap much smaller.

1.4 Unsupervised Image Modeling

As discussed above, limited amount of labeled data is one of the main stopping factors for developing strong general-purpose automatic visual systems. However, practically unlimited amount of raw unlabeled visual data is available in the World Wide Web. This raises a natural question whether unsupervised data can be leveraged in situations when not enough labeled data is available.

Recently this question attracted a lot of attention in the research community. It was demonstrated by many research papers [40, 103, 155, 91] that models, which learn image/video structure from unsupervised visual data, also learn meaningful semantic representations. Models for image modeling were also shown to be useful in reinforcement learning for improving performance of exploration strategies [9, 98]. Furthermore, strong image modeling techniques can be almost directly applied to many standard vision tasks, such as automatic colorization, image restoration, deblurring, super-resolution, etc. Overall, there is a strong evidence that unsupervised image modeling techniques have many important applications.

Historically, unsupervised image modeling is a research area of long tradition that has attracted interest from many different disciplines [120, 96, 57]. However, because of the difficulty of the problem, until recently all existing models were restricted to small image patches, typically between 3x3 and 9x9 pixels, and reflected only low-order statistics, such as edge frequency and orientation [174, 115, 16, 175]. Utilized as prior probabilities in combination with other probabilistic models, such as Markov random fields [44], these models proved to be useful for low-level imaging tasks, such as image denoising and inpainting. Their expressive power is limited, though, as one can see from the fact that samples from the modeled distribution do not resemble natural images, but rather structured noise.

This situation has changed with appearance of powerful and tractable models based on the deep neural networks. In the following we review three widely adopted approaches for image

modeling.

Variational autoencoder or (VAE) is an unsupervised probabilistic model, which was recently proposed and analyzed in [68]. The high-level idea behind VAE is to introduce a random multi-dimensional latent variable $z \in \mathcal{Z}$ and assume that there exists a probabilistic map $f_\theta : \mathcal{Z} \to \mathcal{X}$, parametrized by a vector θ, such that with high probability, for any random z, $f_\theta(z)$ outputs a natural image $X \in \mathcal{X}$. Formally, this means that the distribution p over natural images is modeled in the following way:

$$p_\theta(X) = \int_z p_\theta(X|z)p(z)dz. \tag{1.10}$$

In the above expectation $p_\theta(X|z)$ represents a conditional probability distribution induced by $f_\theta(z)$ and $p(z)$ is a prior over the latent variable z.

Unfortunately, high-dimensional integral in Equation (1.10) is computationally intractable and, thus, computing likelihood $p_\theta(X)$ or learning optimal parameters θ, which maximize the likelihood of training data, is practically unfeasible.

In order to circumvent this obstacle the framework of VAEs approximates the true log-likelihood using the following lower bound:

$$\log p_\theta(X) \geq \mathbb{E}_{q_w(z|X)} \log p_\theta(X|z) - \mathrm{KL}(q_w(z|X)||p(z)). \tag{1.11}$$

This lower bound holds uniformly for any choice of the prior distribution $p(z)$ and for any choice of conditional distributions $p_\theta(X|z)$ and $q_w(z|X)$. The common choice for $p(z)$ is a separable Gaussian distribution, while $p_\theta(X|z)$ and $q_w(z|X)$ are parametrized by deep neural networks. In this case the bound (1.11) is computationally tractable and by maximizing it with respect to θ and w for all training images the learner automatically maximizes the guarantee on the minimal likelihood of the training data. Moreover, by sampling z from the prior distribution $p(z)$ and then sampling image X from $p_\theta(X|z)$ the learner can sample new images.

Generative adversarial networks or (GANs) were recently proposed by [46] and fall into a category of implicit probabilistic models. The original derivation of GANs relies on formulating the image modeling task using the game theoretical framework.

As in previous section, we first introduce a multidimensional latent variable z distributed according to $p(z)$, where $p(z)$ is typically a separable Gaussian distribution. The GAN framework defines two players, a generator $G_w : z \to \mathcal{X}$ and a discriminator $D_\theta : \mathcal{X} \to [0, 1]$, which are represented as deep neural networks parametrized by vectors w and θ respectively. The generator takes random noise z as input and outputs a natural image X, while the discriminator randomly takes real training image or generated image as input and outputs a probability score that the input image is real. Intuitively, the generator strives to fool the discriminator by producing images, which are indistinguishable from the real images. Generator, in its turn, strives to

learn how to distinguish real images from generated ones. As both networks progress together, the generator gets better at generating natural images, which are hard to distinguish from the real images.

Formally, a solution of this game can be obtained as a saddle point of the following optimization problem:

$$\min_G \max_D \mathbb{E}_{x\sim p(x)} \log D(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D(G(z))) \tag{1.12}$$

However, the GAN framework has multiple well-known drawbacks. First, solving the game is theoretically and practically challenging and often results in unstable optimization process. Second, even though a GAN model can be used to sample new images, it does not provide an explicit likelihood function, which is often desirable in practice. Finally, GANs are susceptible to the mode collapse problem [4], which means that the diversity of images, which can be potentially generated by a trained generator, is much smaller in comparison to the diversity of the true image distribution.

Autoregressive models use the chain rule to decompose a computationally demanding image likelihood function into a product of one dimensional conditional distributions, i.e.

$$p(X) = p(x_1) \prod_{i=2}^{n} p(x_i | x_1, x_2, \ldots, x_{i-1}) \tag{1.13}$$

where lowercase x represents individual pixel intensity values and n is a total number of pixels.

This decomposition remains computationally demanding in a general case, as the number of pixels n is high and each conditional distribution is hard-to-model and, thus, requires complex model (e.g. deep neural network) to be adequately represented. However, recent work [142] makes a couple of important observations, which allow to implement the model (1.13) in a computationally efficient way. First, authors note that, assuming translation invariance of natural images, all conditional distributions from (1.13) can be modeled by a single function. This observation results in a drastic reduction of a number of free parameters. Second, they construct a specialized deep convolutional neural network, which computes all conditional distribution in a single forward pass. This constructions allows to leverage massively parallel hardware and readily available software for fast computation of all terms from (1.13).

Subsequently, the design of the autoregressive models was further improved. In [141] authors propose to use gating non-linearity as the activation function in a deep neural network, demonstrating that it improves modeling performance. They also revise layer connectivity structure and propose a novel architecture, which does not suffer from the blind spot present in the original autoregressive network architecture from [142]. Later, [123] introduces a new likelihood function, which drastically speeds up the convergence of a training procedure.

1.5 Thesis Structure and Contributions

This thesis tackles the problem of high demand for supervised data by modern computer vision models.

In **Chapter 2** we investigate the semantic image segmentation task and derive an improved model, which can learn from weak supervision in a form of global image-level labels and outperforms previously proposed techniques under the same experimental conditions. Specifically, we introduce a new loss function for the weakly-supervised training of semantic image segmentation models based on three guiding principles: to seed with weak localization cues, to expand objects based on the information about which classes can occur in an image, and to constrain the segmentations to coincide with object boundaries. We evaluate our model by training a deep convolutional neural network using the proposed loss function and data from the challenging PASCAL VOC 2012 dataset. We furthermore give insight into the working mechanism of our method by a detailed experimental study that illustrates how the segmentation quality is affected by each term of the proposed loss function as well as their combinations. *This chapter is based on the material from the ECCV 2016 paper [71], which is a joint work with Christoph Lampert.*

In a subsequent **Chapter 3** we further analyze weakly-supervised segmentation methods and observe that they tend to fail for object classes that consistently co-occur with the same background elements, e.g. trains on tracks. We propose a method to overcome these failures by adding a very small amount of model-specific additional annotation. The main idea is to cluster a deep network's mid-level representations and assign object or distractor labels to each cluster. Experiments show substantially improved localization results on the challenging ILSVC2014 dataset for bounding box detection and the PASCAL VOC2012 dataset for semantic segmentation. *This chapter is based on the material from the BMVC 2016 paper [70], which is a joint work with Christoph Lampert.*

In a sequel of the thesis we focus on unsupervised modeling techniques for visual data. In **Chapter 4** we extend the autoregressive family of PixelCNN architectures [142] by incorporating auxiliary variables. Subsequently, we describe two new generative image models that exploit different image transformations as auxiliary variables: a quantized grayscale view of the image or a multi-resolution image pyramid. The proposed models tackle two known shortcomings of existing PixelCNN models: 1) their tendency to focus on low-level image details, while largely ignoring high-level image information, such as object shapes, and 2) their computationally costly procedure for image sampling. We experimentally demonstrate benefits of the proposed models, in particular showing that they produce much more realistically looking image samples than previous state-of-the-art probabilistic models. *This chapter is based on the material from the ICML 2017 paper [73], which is a joint work with Christoph Lampert.*

Chapter 5 is dedicated to the automatic image colorization task. We demonstrate that image modeling techniques from the previous chapter are applicable for this task and derive the state-of-the-art probabilistic colorization model. In particular, our model is able to produce multiple plausible and vivid colorizations for a given grayscale image and is one of the first colorization models to provide a proper stochastic sampling scheme. Moreover, our training procedure is supported by a rigorous theoretical framework that does not require any ad hoc heuristics and allows for efficient modeling and learning of the joint pixel color distribution. We demonstrate strong quantitative and qualitative experimental results on the CIFAR-10 dataset and the challenging ILSVRC 2012 dataset. *This chapter is based on the material from the BMVC 2017 paper [119], which is a joint work with Amélie Royer and Christoph Lampert.*

During my PhD I have also published other papers, which fall out of scope of this thesis. These include a joint work with Matthieu Guillaumin, Vittorio Ferrari and Christoph Lampert on large-scale conditional random fields training [69] published at ECCV 2014, a joint work with Sylvestre-Alvise Rebuffi and Christoph Lampert on incremental learning [111] published at CVPR 2017 and a joint work with Harald Ringbauer, David Field and Nicholas Barton [114] published in the GENETICS journal.

2. Semantic Image Segmentation with Image-Level Labels

2.1 Introduction

In this chapter we focus on the task of semantic image segmentation. Image segmentation is a prominent example of an important vision task, for which creating annotations is especially costly: as reported in [121, 8], manually producing segmentation masks requires at least several worker-minutes per image. Therefore, a large body of previous research studies how to train segmentation models from weaker forms of annotation. It was recently demonstrated [29] that very competitive segmentation models can be trained without full segmentations masks from only object's bounding boxes. However, bounding boxes still require significant amount of work to produce. Alternative and much faster to produce form of weak supervision is per-image labels. Unfortunately, there is currently still a large performance gap between models trained from per-image labels and models trained from full segmentations masks. In this chapter we develop a technique that makes significant progress in closing this gap.

We propose a new composite loss function for training convolutional neural networks for the task of weakly-supervised image segmentation when only per-image labels are available. Our approach relies on the following three insights:

- Image classification neural networks, such as AlexNet [77] or VGG [134], can be used to generate reliable object localization cues (**seeds**), but fail to predict the exact spatial extent of the objects. We incorporate this aspect by using a **seeding loss** that encourages a segmentation network to match localization cues but that is agnostic about the rest of the image.

- To train a segmentation network from per-image annotation, a global pooling layer can be used that aggregates segmentation masks into image-level label scores. The choice of this layer has large impact on the quality of segmentations. For example, max-pooling tends to underestimate the size of objects while average-pooling tends to overestimate it [105]. We propose a **global weighted rank pooling** that is leveraged by **expansion**

loss to expand the object seeds to regions of a reasonable size. It generalizes max-pooling and average pooling and outperforms them in our empirical study.

- Networks trained from image-level labels rarely capture the precise boundaries of objects in an image. This happens due to a lack of fully annotated image masks, which carry precise information about object boundaries. This problem can be addressed partially by using fully-connected conditional random fields (CRF) at test time [], However, this typically leads to unsatisfactory results, as neural network is too confident about misclassified regions, so CRF is not able to correct its errors. We propose a new **constrain-to-boundary loss** that alleviates the problem of imprecise boundaries already at training time. It strives to **constrain** predicted segmentation masks to respect low-level image information, in particular object boundaries.

We name our approach **SEC**, as it is based on three principles: **S**eed, **E**xpand and **C**onstrain. We formally define and discuss the individual components of the SEC loss function in Section 2.3. In Section 2.4 we experimentally evaluate it on the PASCAL VOC 2012 image segmentation benchmark, showing that it substantially outperforms the previous state-of-the-art techniques under the same experimental settings. We also provide further insight by discussing and evaluating the effect of each of our contributions separately through additional experiments.

2.2 Related work

Semantic image segmentation, i.e. assigning a semantic class label to each pixel of an image, is a topic of relatively recent interest in computer vision research, as it required the availability of modern machine learning techniques, such as discriminative classifiers [130, 17] or probabilistic graphical models [109, 93]. As the creation of fully annotated training data poses a major bottleneck to the further improvement of these systems, weakly supervised training methods were soon proposed in order to save annotation effort. In particular, competitive methods were developed that only require partial segmentations [147, 55] or object bounding boxes [83, 173, 29] as training data.

A remaining challenge is, however, to learn segmentation models from just image-level labels [145, 146]. Existing approaches fall into three broad categories. *Graph-based models* infer labels for segments or superpixels based on their similarity within or between images [166, 167, 165, 161, 107]. Variants of *multiple instance learning* [2] train with a per-image loss function, while internally maintaining a spatial representation of the image that can be used to produce segmentation masks [148, 149, 150]. Methods in the tradition of *self-training* [126] train a fully-supervised model but create the necessary pixel-level annotation using the model

itself in an EM-like procedure [162, 163, 169]. Our SEC approach contains aspects of the latter two approaches, as it makes use of a per-image loss as well as per-pixel loss terms.

In terms of segmentation quality, currently only methods based on deep convolutional networks [77, 134] are strong enough to tackle segmentation datasets of difficulty similar to what fully-supervised methods can handle, such as the PASCAL VOC 2012 [42], which we make use of in this work. In particular, *MIL-FCN* [104], *MIL-ILP* [105] and the approaches of [8, 75] leverage deep networks in a multiple instance learning setting, differing mainly in their pooling strategies, i.e. how they convert their internal spatial representation to per-image labels. *EM-Adapt* [100] and *CCNN* [102] rely on the self-training framework and differ in how they enforce the consistency between the per-image annotation and the predicted segmentation masks. *SN_B* [158] adds additional steps for creating and combining multiple object proposals. As far as possible, we provide an experimental comparison to these methods in Section 2.4.

2.3 Weakly supervised segmentation from image-level labels

In this section we present a technical description of our approach. We denote the space of images by $\mathcal{X}$. For any image $X \in \mathcal{X}$, a segmentation mask Y is a collection, $(y_1, \ldots, y_n)$, of semantic labels at n spatial locations. The semantic labels belong to a set $\mathcal{C} = \mathcal{C}' \cup \{c^{\text{bg}}\}$ of size k, where $\mathcal{C}'$ is a set of all foreground labels and c^{bg} is a background label. We assume that the training data, $\mathcal{D} = \{(X_i, T_i)\}_{i=1}^N$, consists of N images, $X_i \in \mathcal{X}$, where each image is weakly annotated by a set, $T_i \subset \mathcal{C}'$, of foreground labels that occur in the image. Our goal is to train a deep convolutional neural network $f(X;\theta)$, parameterized by θ, that models the conditional probability of observing any label $c \in \mathcal{C}$ at any location $u \in \{1, 2, \ldots, n\}$, i.e. $f_{u,c}(X;\theta) = p(y_u = c|X)$. For brevity we will often omit the parameters θ in our notation and write $f(X;\theta)$ simply as $f(X)$.

2.3.1 The SEC loss for weakly supervised image segmentation

Our approach for learning the parameters, θ, of the segmentation neural network relies on minimizing a loss function that has three terms. The first term, L_{seed}, provides localization hints to the network, the second term, L_{expand}, penalizes the network for predicting segmentation masks with too small or wrong objects, and the third term, $L_{\text{constrain}}$, encourages segmentations that respect the spatial and color structure of the images. Overall, we propose to solve the following optimization problem for parameter learning:

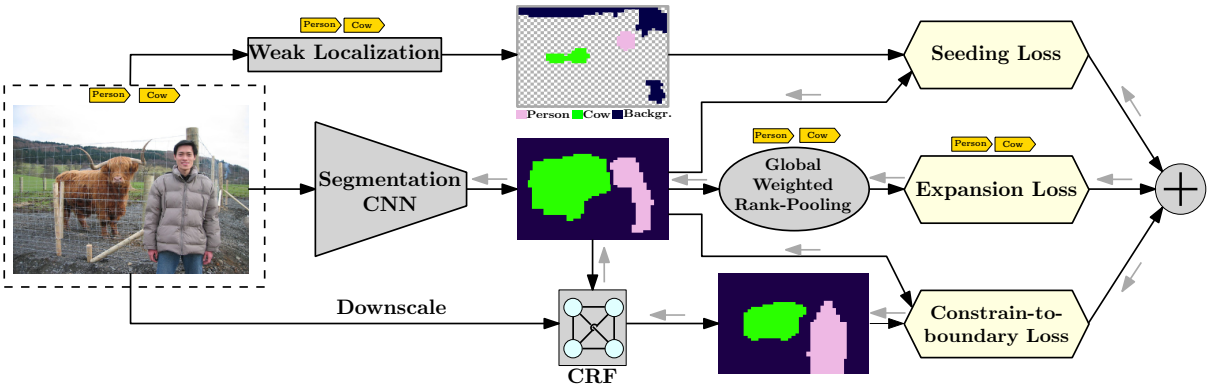


Figure 2.1: A schematic illustration of SEC that is based on minimizing a composite loss function consisting of three terms: *seeding loss*, *expansion loss* and *constrain-to-boundary loss*. See Section 2.3 for details.

$$\min_{\theta} \sum_{(X,T)\in\mathcal{D}} \left[L_{\text{seed}}(f(X;\theta),T) + L_{\text{expand}}(f(X;\theta),T) + L_{\text{constrain}}(X,f(X;\theta))\right]. \tag{2.1}$$

In the rest of this section we explain each loss term in detail. A schematic overview of the setup can be found in Figure 2.1.

Seeding loss with localization cues.

Image-level labels do not explicitly provide any information about the position of semantic objects in an image. Nevertheless, as was noted in many recent research papers [97, 171, 133, 7], deep image classification networks that were trained just from image-level labels, may be successfully employed to retrieve cues on object localization. We call this procedure *weak localization* and illustrate it in Figure 2.2.

Unfortunately, localization cues typically are not precise enough to be used as full and accurate segmentation masks. However, these cues can be very useful to guide the weakly-supervised segmentation network. We propose to use a *seeding loss* to encourage predictions of the neural network to match only "landmarks" given by the weak localization procedure while ignoring the rest of the image. Suppose that S_c is a set of locations that are labeled with class c by the weak localization procedure. Then, the *seeding loss* L_{seed} has the following form:

$$L_{\text{seed}}(f(X),T,S_c) = -\frac{1}{\sum_{c\in T}|S_c|}\sum_{c\in T}\sum_{u\in S_c}\log f_{u,c}(X). \tag{2.2}$$

Note that for computing L_{seed} one needs the weak localization sets, S_c, so that many existing techniques from the literature can be used, essentially, as *black boxes*. In this work, we rely on [171] for weakly localizing foreground classes. However, this method does not provide a

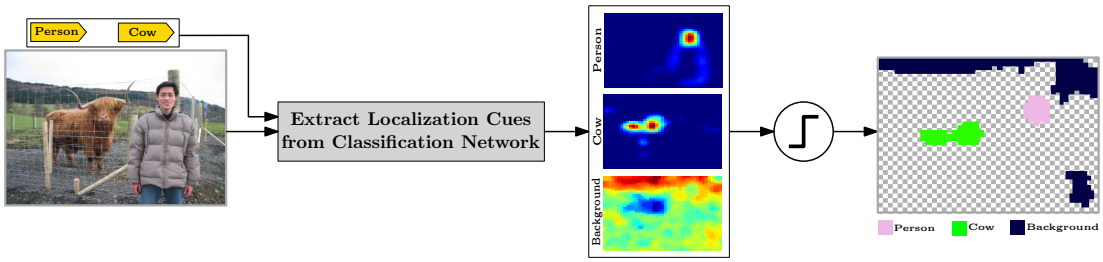


Figure 2.2: The schematic illustration of the weak localization procedure.

direct way to select confident background regions, therefore we use the gradient-based saliency detection method from [133] for this purpose. We provide more details on the weak localization procedure in Section 2.4.

Expansion loss with global weighted rank pooling.

To measure if a segmentation mask is consistent with the image-level labels one can aggregate segmentation scores into classification scores and apply the standard loss function for multi-label image classification. In the context of weakly-supervised segmentation/detection various techniques were used by researches to aggregate score maps into a classification scores. The most prominent ones are *global max-poling* (GMP) [97] that assigns any class c in any image X a score of $\max\limits_{u\in\{1,\dots,n\}} f_{u,c}(X)$ and *global average-pooling* [171] that assigns it a score of $\frac{1}{n}\sum\limits_{u=1}^{n} f_{u,c}(X)$.

Both ways of aggregation have been successfully used in practice. However, they have their own drawbacks. For classes which are present in an image GMP only encourages the response for a single location to be high, while GAP encourages all responses to be high. Therefore, GMP results in a segmentation network that often underestimates the sizes of objects, while network trained using GAP, in contrast, often overestimates them. Our experiments in Section 2.4 support this claim empirically.

In order to overcome these drawbacks we propose a *global weighted rank-pooling (GWRP)*, a new aggregation technique, which can be seen as a generalization of GMP and GAP. GWRP computes a weighted average score for each class, where weights are higher for more promising locations. This way it encourages objects to occupy a certain fraction of an image, but, unlike GAP, is less prone to overestimating object sizes.

Formally, let an index set $I^c = \{i_1, \dots, i_n\}$ define the descending order of prediction scores for any class $c \in \mathcal{C}$, i.e. $f_{i_1,c}(x) \geq f_{i_2,c}(x) \geq \cdots \geq f_{i_n,c}(x)$ and let $0 < d_c <= 1$ be a decay parameter for class c. Then we define the GWRP classification scores, $G_c(f(X), d_c)$, for an

image X, as following:

$$G_c(f(X); d_c) = \frac{1}{Z(d_c)} \sum_{j=1}^{n} (d_c)^{j-1} f_{i_j,c}(X), \text{ where } Z(d_c) = \sum_{j=1}^{n} (d_c)^{j-1}. \tag{2.3}$$

Note, that for $d_c = 0$ GWRP turns into GMP (adopting the convention that $0^0 = 1$), and for $d_c = 1$ it is identical to GAP. Therefore, GWRP generalizes both approaches and the decay parameter can be used to interpolate between the behavior of both extremes.

In principle, the decay parameter could be set individually for each class and each image. However, this would need prior knowledge about how large objects of each class typically are, which is not available in the weakly supervised setting. Therefore, we only distinguish between three groups: for object classes that occur in an image we use a decay parameter d_+, for object classes that do not occur we use d_-, and for background we use d_{bg}. We will discuss how to choose their values in Section 2.4.

In summary, the *expansion loss* term is

$$\begin{aligned} L_{\text{expand}}(f(X), T) = &- \frac{1}{|T|} \sum_{c \in T} \log G_c(f(X); d_+) \\ &- \frac{1}{|\mathcal{C}' \backslash T|} \sum_{c \in \mathcal{C}' \backslash T} \log(1 - G_c(f(X); d_-)) - \log G_{c^{\text{bg}}}(f(X); d_{\text{bg}}). \end{aligned} \tag{2.4}$$

Constrain-to-boundary loss.

The high level idea of the *constrain-to-boundary loss* is to penalize the neural network for producing segmentations that are discontinuous with respect to spatial and color information in the input image. Thereby, it encourages the network to learn to produce segmentation masks that match up with object boundaries.

Specifically, we construct a fully-connected CRF, $Q(X, f(X))$, as in [74], with unary potentials given by the logarithm of the probability scores predicted by the segmentation network, and pairwise potentials of fixed parametric form that depend only on the image pixels. We downscale the image X, so that it matches the resolution of the segmentation mask, produced by the network. More details about the choice of the CRF parameters are given in Section 2.4. We then define the *constrain-to-boundary loss* as the mean KL-divergence between the outputs of the network and the outputs of the CRF, i.e.:

$$L_{\text{constrain}}(X, f(X)) = \frac{1}{n} \sum_{u=1}^{n} \sum_{c \in \mathcal{C}} Q_{u,c}(X, f(X)) \log \frac{Q_{u,c}(X, f(X))}{f_{u,c}(X)}. \tag{2.5}$$

This construction achieves the desired effect, since it encourages the network output to coincide with the CRF output, which itself is known to produce segmentation that respect image boundaries. An illustration of this effect can be seen in Figure 2.1.

2.3.2 Training

The proposed network can be trained in an end-to-end way using back-propagation, provided that the individual gradients of all layers are available. For computing gradients of the fully-connected CRF we employ the procedure from [139], which was successfully used in the context of semantic image segmentation. Figure 2.1 illustrates the flow of gradients for the back-propagation procedure with gray arrows.

2.4 Experiments

In this section we validate our proposed loss function experimentally, including a detailed study of the effects of its different terms.

2.4.1 Experimental setup

Dataset and evaluation metric. We evaluate our method on the PASCAL VOC 2012 image segmentation benchmark, which has 21 semantic classes, including background [42]. The dataset images are split into three parts: training (*train*, 1464 images), validation (*val*, 1449 images) and testing (*test*, 1456 images). Following the common practice we augment the training part by additional images from [53]. The resulting *trainaug* set has 10,582 weakly annotated images that we use to train our models. We compare our approach with other approaches on both *val* and *test* parts. For the *val* part, ground truth segmentation masks are available, so we can evaluate results of different experiments. We therefore use this data also to provide a detailed study of the influence of the different components in our approach. The ground truth segmentation masks for the *test* part are not publicly available, so we use the official PASCAL VOC evaluation server to obtain quantitative results. As evaluation measure we use the standard PASCAL VOC 2012 segmentation metric: mean intersection-over-union (mIoU).

Segmentation network. As a particular choice for the segmentation architecture, we use *DeepLab-CRF-LargeFOV* from [21], which is a slightly modified version of the 16-layer VGG network [134]. The network has inputs of size 321x321 and produces segmentation masks of size 41x41, see [21] for more details on the architecture. We initialize the weights for the last (prediction) layer randomly from a normal distribution with mean 0 and variance 0.01. All other convolutional layers are initialized from the publicly available VGG model [134]. Note, that in principle, our loss function can be combined with any deep convolutional neural network.

Localization networks. The localization networks for the foreground classes and the background class are also derived from the standard VGG architecture. In order to improve the localization performance, we finetune these networks for solving a multilabel classification problem

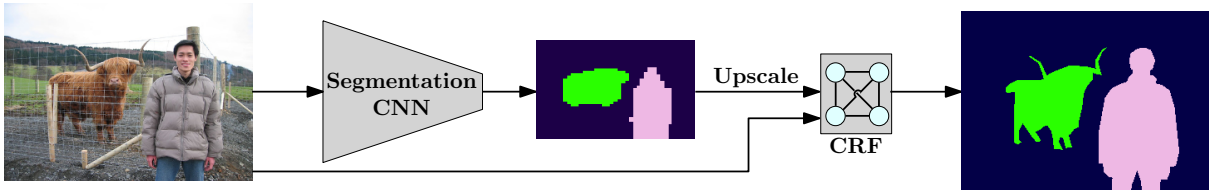


Figure 2.3: The schematic illustration of our approach at test time.

on the *trainaug* data. Due to space limitations we provide exact details on these networks and optimization parameters in the technical report [71].

Note, that in order to reduce the computational effort and memory consumption required for training SEC it is possible to precompute the localization cues. If precomputed cues are available SEC imposes no additional overhead for evaluating and storing the localization networks at training time.

Optimization. For training the network we use the batched stochastic gradient descent (SGD) with parameters used successfully in [21]. We run SGD for 8000 iterations, the batch size is 15 (reduced from 30 to allow simultaneous training of two networks), the dropout rate is 0.5 and the weight decay parameter is 0.0005. The initial learning rate is 0.001 and it is decreased by a factor of 10 every 2000 iterations. Overall, training on a `GeForce TITAN-X` GPU takes 7-8 hours, which is comparable to training times of other models, reported, e.g., in [100, 102].

Decay parameters. The GWRP aggregation requires specifying the decay parameters, d_{-}, d_{+} and d_{bg}, that control the weights for aggregating the scores produced by the network. Inspired by the previous research [100, 102] we do so using the following rules-of-thumb that express prior beliefs about natural images:

- for semantic classes that are not present in the image we want to predict as few pixels as possible. Therefore, we set $d_{-} = 0$, which corresponds to GMP.
- for semantic classes that are present in the image we suggest that the top 10% scores represent 50% of the overall aggregated score. For our 41x41 masks this roughly corresponds to $d_{+} = 0.996$.
- for the background we suggest that the top 30% scores represent 50% of the overall aggregated score, resulting in $d_{\mathrm{bg}} = 0.999$.

Fully-connected CRF at training time. In order to enforce the segmentation network to respect the boundaries of objects already at training time we use a fully-connected CRF [74]. As parameters for the pairwise interactions, we use the default values from the authors' public implementation, except that we multiply all spatial distance terms by 12 to reflect the fact that

we downscaled the original image in order to match the size of the predicted segmentation mask.

Inference at test time. Our segmentation neural network is trained to produce probability scores for all classes and locations, but the spatial resolution of a predicted segmentation mask is lower than the original image. Thus, we upscale the predicted segmentation mask to match the size of the input image, and then apply a fully-connected CRF [74] to refine the segmentation. This is a common practice, which was previously employed, e.g., in [100, 102, 21]. Figure 2.3 shows a schematic illustration of our inference procedure at test time.

Reproducibility. In our experiments we rely on the *caffe* deep learning framework [61] in combination with a *python* implementation of the SEC loss. The code and pretrained models are publicly available[1].

2.4.2 Results

Quantitative Results. Table 2.1 compares the performance of our weakly supervised approach with previous approaches that are trained in the same setup, i.e. using only images from PASCAL VOC 2012 and only image-level labels. It shows that SEC substantially outperforms the previous techniques. On the *test* data, where the evaluation is performed by an independent third party, the PASCAL VOC evaluation server, it achieves 13.5% higher mean intersection-over-union score than the previously proposed approaches with new best scores on 20 out of 21 semantic classes. On the validation data, for which researchers can compute scores themselves, SEC improves over the previous techniques by 14.1%, and achieves new best scores on 19 out of the 21 classes.

Results of other weakly-supervised methods on PASCAL VOC and the fully-supervised variant of DeepLab are summarized in Table 2.2. We provide these results for reference but emphasize that they should not simply be compared to Table 2.1, because the underlying methods were trained on different (and larger) training sets or were given additional forms of weak supervision, e.g. user clicks. Some entries need further explanation in this regard: [100] reports results for the EM-Adapt model when trained with weak annotation for multiple image crops. The same model was reimplemented and trained with only per-image supervision in [102], so these are the values we report in Table 2.1. The results reported for SN_B [158] and the *seg* variant of the MIL+ILP+SP [105] are incomparable to others because they were obtained with help of *MCG* region proposals [3] that were trained in a fully supervised way on PASCAL VOC data. Similarly, MIL+ILP+SP-*bb* makes use of bounding box proposals generated by the BING method [24] that was trained using PASCAL VOC bounding box annotation.

[1] https://github.com/kolesman/SEC

PASCAL VOC 2012 *val* set	[8] (Img+Obj)	[65] (stage1)	EM-Adapt (re-impl. of [102])	CCNN [102]	MIL+ILP +SP-sppxl† [105]	SEC (proposed)
background		71.7*	67.2	68.5	77.2	**82.4**
aeroplane		30.7*	29.2	25.5	37.3	**62.9**
bike		**30.5***	17.6	18.0	18.4	26.4
bird		26.3*	28.6	25.4	25.4	**61.6**
boat		20.0*	22.2	20.2	**28.2**	27.6
bottle		24.2*	29.6	36.3	31.9	**38.1**
bus		39.2*	47.0	46.8	41.6	**66.6**
car		33.7*	44.0	47.1	48.1	**62.7**
cat		50.2*	44.2	48.0	50.7	**75.2**
chair		17.1*	14.6	15.8	12.7	**22.1**
cow		29.7*	35.1	37.9	45.7	**53.5**
diningtable		22.5*	24.9	21.0	14.6	**28.3**
dog		41.3*	41.0	44.5	50.9	**65.8**
horse		35.7*	34.8	34.5	44.1	**57.8**
motorbike		43.0*	41.6	46.2	39.2	**62.3**
person		36.0*	32.1	40.7	37.9	**52.5**
plant		29.0*	24.8	30.4	28.3	**32.5**
sheep		34.9*	37.4	36.3	44.0	**62.6**
sofa		23.1*	24.0	22.2	19.6	**32.1**
train		33.2*	38.1	38.8	37.6	**45.4**
tv/monitor		33.2*	31.6	36.9	35.0	**45.3**
average	32.2	33.6*	33.8	35.3	36.6	**50.7**

PASCAL VOC 2012 *test* set	MIL-FCN [104]	CCNN [102]	MIL+ILP +SP-sppxl† [105]	Region score pooling [75]	SEC (proposed)
background		≈71‡	74.7	≈74‡	**83.5**
aeroplane		24.2	38.8	33.1	**56.4**
bike		19.9	19.8	21.7	**28.5**
bird		26.3	27.5	27.7	**64.1**
boat		18.6	21.7	17.7	**23.6**
bottle		38.1	32.8	38.4	**46.5**
bus		51.7	40.0	55.8	**70.6**
car		42.9	50.1	38.3	**58.5**
cat		48.2	47.1	57.9	**71.3**
chair		15.6	7.2	13.6	**23.2**
cow		37.2	44.8	37.4	**54.0**
diningtable		18.3	15.8	**29.2**	28.0
dog		43.0	49.4	43.9	**68.1**
horse		38.2	47.3	39.1	**62.1**
motorbike		52.2	36.6	52.4	**70.0**
person		40.0	36.4	44.4	**55.0**
plant		33.8	24.3	30.2	**38.4**
sheep		36.0	44.5	48.7	**58.0**
sofa		21.6	21.0	26.4	**39.9**
train		33.4	31.5	31.8	**38.4**
tv/monitor		38.3	41.3	36.3	**48.3**
average	25.7	35.6	35.8	38.0	**51.7**

(* results from unpublished/not peer-reviewed manuscripts, † trained on ImageNet, ‡ value inferred from average)

Table 2.1: Results on PASCAL VOC 2012 (mIoU in %) for weakly-supervised semantic segmentation with only per-image labels.

method	*val*	*test*	comments
DeepLab [21]	67.6	70.3	fully supervised training
STC [159]	49.8	51.2	trained on Flickr
TransferNet [56]	52.1	51.2	trained on MS COCO; additional supervision: from segmentation mask of other classes
[8] (1Point)	42.7	–	additional supervision: 1 click per class
[8] (AllPoints-weighted)	43.4	–	additional supervision: 1 click per instance
[8] (squiggle)	49.1	–	additional supervision: 1 squiggle per class
EM-Adapt [100]	38.2	39.6	uses weak labels of multiple image crops
SN_B [158]	41.9	43.2	uses MCG region proposals (see text)
MIP+ILP+SP-seg [105]	42.0	40.6	trained on ImageNet, MCG proposals (see text)
MIL+ILP+SP-bb [105]	37.8	37.0	trained on ImageNet, BING proposals (see text)
Comb. cues [118]	52.8	53.7	follow-up publication
AE-PSL [157]	55.0	55.7	follow-up publication
DCSP-ResNet-101 [20]	60.8	61.9	follow-up publication

Table 2.2: Additional results (mIoU %) on PASCAL VOC 2012

The last three entries from Table 2.2 represent follow-up publications. Notably, all of these publications rely on the idea of using segmentation seeds from a classification network. In [118] authors propose a unified framework which uses a single convolutional neural network for producing segmentation seeds and for predicting final segmentation masks. A paper [157] introduces the idea of adversarial erasing, which improves quality of segmentation seeds by iteratively masking out semantically discriminative image regions. Finally, [20] proposes to additionally employ a separate saliency detection network and achieves the current state-of-the-art segmentation performance on the PASCAL VOC 2012 dataset.

Note that we do include the *sppxl* variant of MIL+ILP+SP in Table 2.1. While it is trained on roughly 760.000 images of the ImageNet dataset, we do not consider this an unfair advantage compared to our and other methods, because those implicitly benefit from ImageNet images as well when using pretrained classification networks for initialization.

Qualitative Results. Figure 2.4 illustrates typical successful segmentations. It shows that our method can produce accurate segmentations even for non-trivial images and recover fine details of the boundary. Figure 2.5 illustrates some failure cases. As is typical for weakly-supervised systems, SEC has problems segmenting objects that occur almost always in front of the same background, e.g. boats on water, or trains on tracks. We addressed this problem recently in follow-up work [70]. A second failure mode is that object regions can be segmented correctly, but assigned wrong class labels. This is actually quite rare for SEC, which we attribute to the fact that the DeepLab network has a large field-of-view and therefore can make use of the full image when assigning labels. Finally, it can also happen that segmentations cover only parts of objects. This is likely due to imperfections of the weak localization cues that tend to reliably detect only the most discriminative parts of an object, e.g. the face of a person. This might not be sufficient to segment the complete object, however, especially when objects overlap each other or consist of multiple components of very different appearance.

2.4.3 Detailed Discussion

To provide additional insight into the working mechanisms of the SEC loss function, we performed two further sets of experiments on the *val* data. First, we analyze different global pooling strategies, and second, we perform an ablation study that illustrates the effect of each of the three terms in the proposed loss function visually as well as numerically.

Effect of global pooling strategies. As discussed before, the quality of segmentations depends on which global pooling strategy is used to convert segmentation mask into per-image classification scores. To quantify this effect, we train three segmentation networks from weak supervision, using either GMP, GAP or GWRP as aggregation methods for classes that are present in the image. For classes that are not present we always use GMP, i.e. we penalize

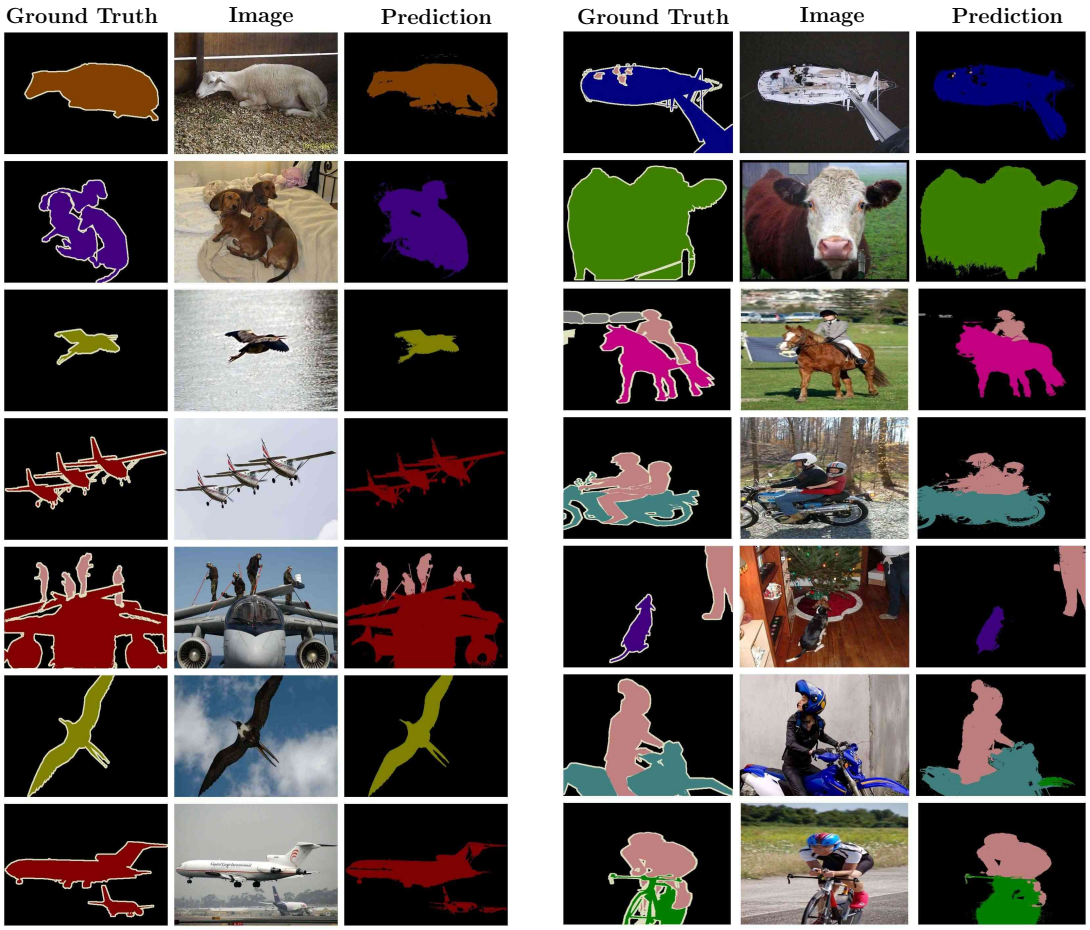


Figure 2.4: Examples of predicted segmentations (*val* set, successfull cases).

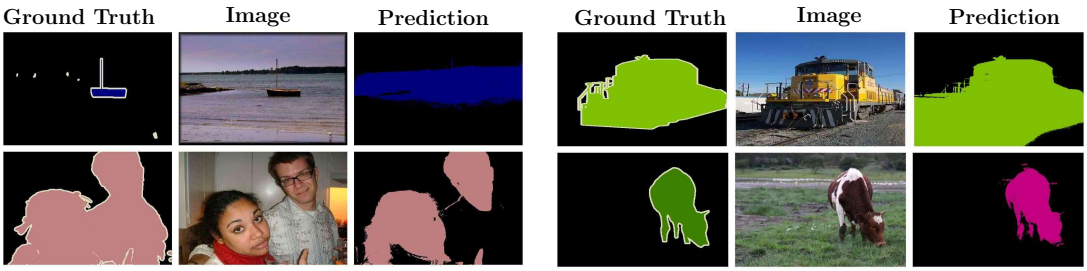
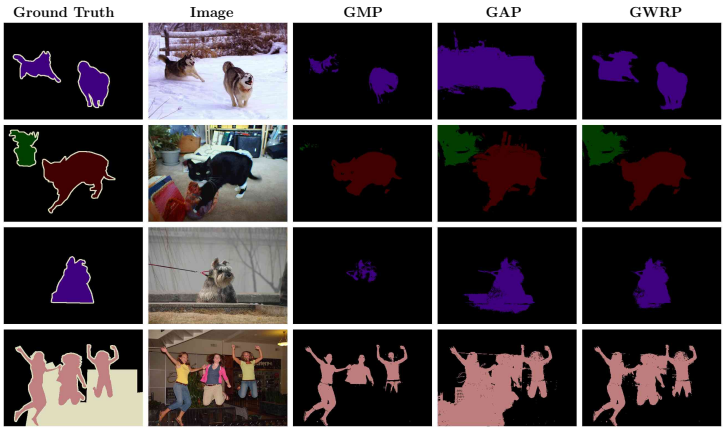


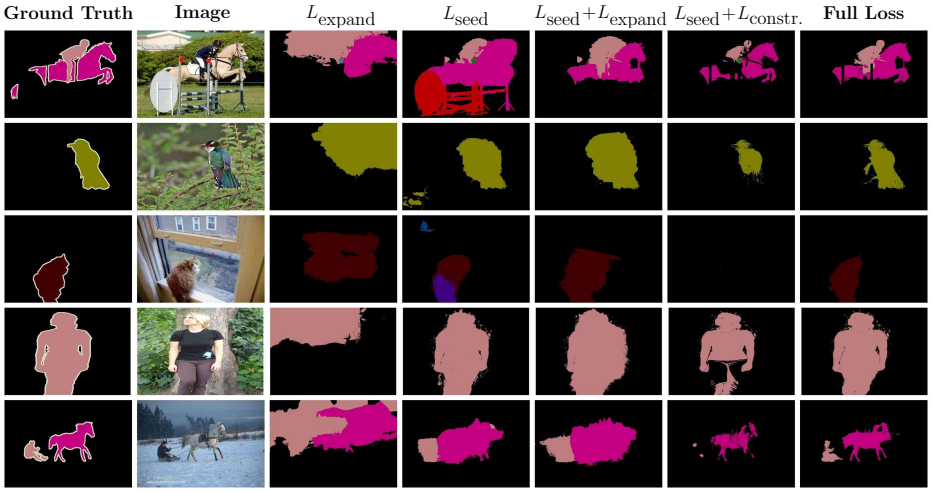
Figure 2.5: Examples of predicted segmentations (*val* set, failure cases).

any occurrence of these classes. In Figure 2.6 we demonstrate visual results for every pooling strategy and report two quantities: the fraction of pixels that are predicted to belong to a foreground (fg) class, and the segmentation performance as measured by mean IoU. We observe that GWRP outperforms the other method in terms of segmentation quality and the fractions of predicted foreground pixels supports our earlier hypothesis: the model trained with GMP tends to underestimate object sizes, while the model trained with with GAP on average overestimates them. In contrast, the model trained with GWRP, produces segmentations in which objects are,



pooling method	fg fraction	mIoU *(val)*
GMP	20.4	46.5
GAP	35.6	45.7
GWRP	25.8	50.7
ground truth	26.7	–

Figure 2.6: Results on the *val* set and examples of segmentation masks for models trained with different pooling strategies.



loss function	mIoU *(val)*
L_{expand}	20.6
L_{seed}	45.4
$L_{\text{seed}} + L_{\text{expand}}$	44.3
$L_{\text{seed}} + L_{\text{constrain}}$	50.4
all terms	50.7

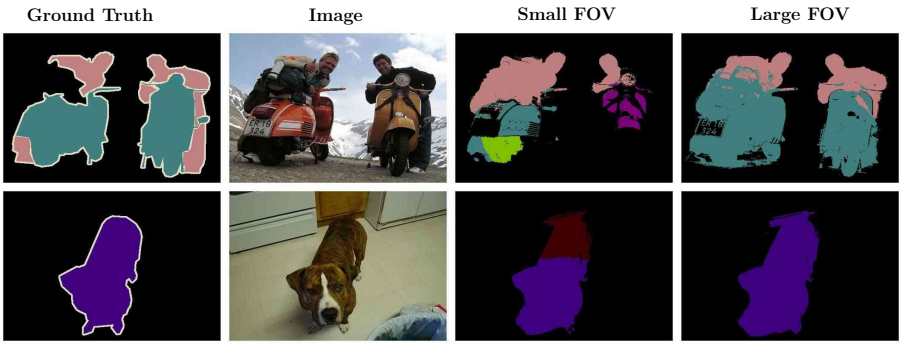
Figure 2.7: Results on the *val* set and examples of segmentation masks for models trained with different loss functions.

on average, close to the correct size[2].

Effect of the different loss terms. To investigate the contribution of each term in our composite loss function we train segmentation networks with loss functions in which different terms of the SEC loss were omitted. Figure 2.7 provides numerical results and illustrates typical segmentation mistakes that occur when certain loss terms are omitted. Best results are achieved when all three loss terms are present. However, the experiments also allow us to draw two interesting additional conclusions about the interaction between the loss terms.

Semi-supervised loss and large field-of-view. First, we observe that having L_{seed} in the loss function is crucial to achieve competitive performance. Without this loss term our segmentation network fails to reflect the localization of objects in its predictions, even though the network

[2] Note that these experiments were done after the network architecture and parameters were fixed. In particular, we did not tune the decay parameters for this effect.



field of view	mIoU *(val)*
211x211	38.1
378x378	50.7

Figure 2.8: Results on the *val* set and examples of segmentation masks for models with small or large field-of-views.

does match the global label statistics rather well. See the third column of Figure 2.7 for the illustration of this effect.

We believe that this effect can be explained by the large (378x378) field-of-view (FOV) of the segmentation network[3]: if an object is present in an image, then the majority of the predicted scores may be influenced by this object, no matter where object is located. This helps in predicting the right class labels, but can negatively affect the localization ability. Other researchers addressed this problem by explicitly changing the architecture of the network in order to reduce its field-of-view [100]. However, networks with a small field-of-view are less powerful and often fail to recognize which semantic labels are present on an image. We conduct an additional experiment (see the technical report [71] for details) that confirm that SEC with a small (211x211) field-of-view network performs clearly worse than with the large (378x378) field-of-view network, see Figure 2.8 for numeric results and visual examples. Thus, we conclude that the seeding loss provides the necessary localization guidance that enables the large field-of-view network to still reliably localize objects.

Effects of the expansion and constrain-to-boundary losses. By construction, the constrain-to-boundary loss encourages nearby regions of similar color to have the same label. However, this is often not enough to turn the weak localization cues into segmentation masks that cover a whole object, especially if the object consists of visually dissimilar parts, such as people wearing clothes of different colors. See the sixth column of Figure 2.7 for an illustration of this effect.

The expansion loss, based on GWRP, suppresses the prediction of classes that are not meant to be in the image, and it encourages classes that are in the image to have reasonable sizes. When combined with the seeding loss, the expansion loss actually results in a drop in performance. The fifth column of Figure 2.7 shows an explanation of this: objects sizes are generally

[3] We report the theoretical fields-of-view inferred from the network architecture. The empirical field-of-view that is actually used by the network can be smaller [172].

increased, but the additionally predicted regions do not match the image boundaries.

In combination, the seeding loss provides reliable seed locations, the expansion loss acts as a force to enlarge the segmentation masks to a reasonable size, and the constrain-to-boundary loss constrains the segmentation mask to line up with image boundaries, thus integrating low-level image information. The result are substantially improved segmentation masks as illustrated in the last column of Figure 2.7.

2.5 Conclusion

We propose a new loss function for training deep segmentation networks when only image-level labels are available. We demonstrate that our approach outperforms previous state-of-the-art methods by a large margin when used under the same experimental conditions and provide a detailed ablation study.

We also identify potential directions that may help to further improve weakly-supervised segmentation performance. Our experiments show that knowledge about object sizes can dramatically improve the segmentation performance. SEC readily allows incorporating size information through decay parameters, but a procedure for estimating object sizes automatically would be desirable [129]. A second way to improve the performance would be stronger segmentation priors, for example about shape or materials. This could offer a way to avoid mistakes that are currently typical for weakly-supervised segmentation networks.

3. Semantic Image Segmentation with Micro-Annotation

3.1 Introduction

In this chapter we aim to further improve performance of weakly-supervised image segmentation techniques. Analyzing the existing methods in this field, we observe that certain object classes, for example *trains* or *boats*, are harder to segment from weak supervision than others, in the sense that there is a big gap between the prediction quality of the models trained from just per-image class labels and the quality achieved by models trained with full supervision.

We hypothesize that weakly-supervised segmentation models tend to fail when object classes systematically co-occur with *distractors* (background or certain other classes), for example *train* and *tracks*. Per-image class annotation simply does not contain necessary information to reliably learn the difference between objects and distrators. In this work, we argue that the best way to overcome this problem is to collect a tiny amount of additional annotation, which we call *micro-annotation*.

Our approach relies on the assumption that even though it might be impossible for a classifier to learn from weak per-image annotation which parts of an image are the object of interest and which are distractors, it will still be possible to distinguish both groups from each other by clustering their appearance in a suitable representation. Then, all we need in order to improve the quality of a weakly-supervised segmentation systems is a way to find out which clusters belong to distractors – which is an easy task for a human annotator – and suppress them.

This micro-annotation approach can be used in combination with many existing weakly-supervised methods. In order to strengthen our experimental evaluation we also conduct numerical evaluation for predicting object bounding boxes. Specifically, in this work we combine micro-annotation technique with the recently proposed competitive methods for weakly-supervised bounding box prediction and for weakly-supervised semantic segmentation, showing improved results on the challenging ILSVR2014 and PASCAL VOC2012 datasets.

Apart from its practical usefulness, an interesting aspect of this approach is that it asks for user annotation after an initial model has already been trained. This allows the requested

information to depend on the original model's output, hopefully with the effect that the new information has a maximally beneficial effect on the prediction quality. This setup resembles *active learning*, but with an even better relation between the amount of annotation and the model improvement. In active learning, the annotation provides information through the labeling of individual images, and each provided label typically influences the model parameters by an amount inversely proportional to the total size of the training set. Consequently, active learning is most beneficial for models trained on small datasets. In our approach, a single user interaction can have a large effect on the model parameters and thereby the prediction quality, namely when it establishes that all detected patterns of a certain type are distractors and should be suppressed. The size of the training set plays no role for this effect, and indeed we observe a significant improvement even for models trained on very large datasets.

3.2 Related work

Many methods for object localization have been proposed that can be trained in a weakly-supervised way from per-image class label annotation. The majority of these methods predict either object bounding boxes [10, 11, 25, 34, 136, 137, 154, 171, 7] or per-pixel segmentation masks [145, 146, 148, 149, 150, 162, 163, 169, 104, 102, 105, 100, 72]. The currenlty most successful approaches obtain localization hints from convolutional neural networks that are trained for the of image classification, e.g. [171, 72]. The micro-annotation method can in principle be used on top of any of such method, as long as that has the ability to produce per-location score maps.

Much fewer works have studied how the process of data annotation can be improved by the power of strong computer vision systems. Two related research directions are *visual recognition with humans in the loop* [13] and *active learning* [127]. In the human in the loop concept, an automatic system and a human user work together during the prediction stage. For example, in a fine-grained classification task [13, 32, 152, 153], the machine would output a selection of possible labels and the human user would pick the most appropriate one. Typically, this process does not improve the model parameters, though, so feedback from a human is always required to make high quality predictions. Active learning also has the goal of harvesting human expertise, but it does so during the training stage: an automatic system has access to a large number of unlabeled images and can ask a human annotator to specifically annotate a subset of them [43, 62, 63, 108]. This procedure can reduce the amount of necessary annotation, e.g. when the system only ask for annotation of images that it is not already certain about anyway. In the context of object localization, a more efficient weakly-supervised variant of active learning has been proposed in which a human user only has to annotate if a predicted bounding box is correct or not, instead of having to draw it manually [99]. Our approach differs from

these setting in particular in the fact that we do not ask a user to provide feedback about individual images, but about clusters in the learned data representation, which reflects information extracted from the whole training set. Thereby, we require much less interaction with the annotator, namely in the order of the number of classes instead of in the order of the number of training images.

On the technical level, our method is related to recent approaches for discovering object detectors in deep convolutional neural networks [81, 172, 132]. However, these rely on the assumption that part-detectors correspond to individual convolution filter outputs, whereas our clustering approach finds co-occurring patterns in the distributed representation learned by the network. The fact that our method identifies image structures by clustering visual representations across many images resembles image co-segmentation [117]. It differs, however, from these earlier works in how it organizes the clustering process and how it uses the structure that are found.

3.3 Micro-Annotation Technique

In this section we formally introduce the proposed procedure for collecting micro-annotation (illustrated in Figure 3.1) and improving object localization (illustrated in Figure 3.3). The main steps for obtaining the additional annotation for each class are: (i) represent all predicted foreground regions of all images by feature vectors, (ii) cluster the feature vectors (iii) visualize the clusters and let an annotator select which ones actually corresponds to the object class of interest. The information about clusters and their annotation can then be used to better localize objects: (iv) for any (new) image, predict a foreground map using only the image regions that match clusters labeled as 'object'.

In the rest of the section we explain these steps in detail. For this, we denote the set of training images by $\mathcal{D}$ and assume a fixed set of semantic categories $\mathcal{Y}$ that we want to localize. The subsequent construction can be performed independently for each object class. By $y \in \mathcal{Y}$ we always denote the current class of interest.

We assume that we are given a pretrained deep convolutional neural network, f, that predicts the presence of semantic categories for an input image X, but that can also be leveraged to predict the spatial location of semantic objects in input images. Formally, we assume that each image is regularly split into a set of non-overlaping rectangular regions, $\mathcal{U}$, and that f gives rise to a *scoring function* that assigns a localization score, $S^y_u(X)$, to each image region $u \in \mathcal{U}$. Furthermore, we assume the availability of a *thresholding procedure* that converts *score maps*, $S^y(X) \in \mathbb{R}^{|\mathcal{U}|}$, to a set of image regions, $D^y(X) \subset \mathcal{U}$, that represent the predicted localization of the class y. We discuss particular choices for the functions S and D, in Section 3.4.

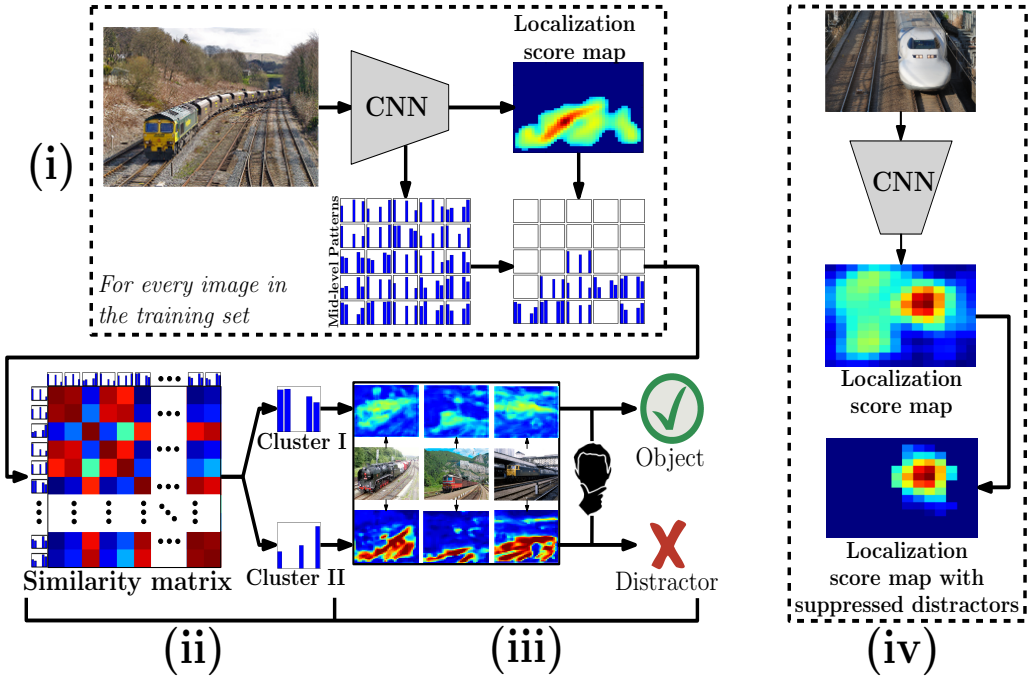


Figure 3.1: Schematic illustration of the micro-annotation approach: (i) for every image region across all training images that is predicted to show the object of interest (here: *train*), compute its mid-level feature representation (pattern) and form a pool of patterns, (ii) find characteristic clusters in this pool, (iii) visualize the clusters by heatmaps and ask a user to annotate them as representing either the object or a distractor and (iv) at test time modify the produced localization maps by discarding regions that correspond to the distractors.

(i) Region representations. At any fixed layer of the deep network we can form a feature vector, $\phi_u(X) \in \mathbb{R}^k$, (called a *pattern*) for any region, $u \in \mathcal{U}$, by concatenating the real-valued convolutional filter activations from the fixed layer. For simpler use in the clustering step we assume that $\phi_u(X)$ has nonnegative entries, e.g. after a ReLU operation, and that it is L^2-normalized. These are not principled restrictions, though. Arbitrary features could be used in combination with a different clustering algorithm. We form a *pattern set* $A^y = \{\phi_u(X) | \forall u \in D^y(X), \forall X \in \mathcal{D}\}$, i.e. the features of all image regions with predicted label y.

(ii) Clustering. We partition the set A^y into a group of clusters, $P^y = \{C_1, \ldots, C_k\}$, by *spectral clustering* [151]. The number of clusters is determined automatically using the fact that the k-th eigenvalues of the graph Laplacian (computed during the clustering) reflects the quality of creating k clusters (Algorithm 3, line 5).

General spectral clustering does not scale well to large datasets, because when used with a generic similarity measure it requires memory quadratic and runtime cubic in the number of patterns. This is not the case for us: we use a linear (inner product) similarity measure between patterns, which allows us to avoid storing the quadratically sized similarity matrix *(line 1)* explicitly. The necessary eigenvalue problem *(line 4)* we solve efficiently by the Lanczos

Algorithm 3: Spectral clustering algorithm

input : patterns A^y, eigenvalue threshold ρ (default: 0.7), lower bound m (default: 2) and upper bound M (default 4) for the number of clusters

1 Compute the similarity matrix: $W^y \in \mathbb{R}^{|A^y| \times |A^y|}$ with $W^y_{a,b} = \langle a, b \rangle \geq 0$ for all $a, b \in A^y$.

2 Compute the diagonal matrix: $D^y \in \mathbb{R}^{|A^y| \times |A^y|}$ with $D^y_{a,a} = \sum_{b \in A^y} W^y_{a,b}$ for all $a \in A^y$.

3 Compute the Laplacian matrix: $L^y = D^y - W^y$.

4 Compute the M smallest eigenvalues $\{\lambda_1, \ldots \lambda_M\}$ and eigenvectors $\{v_1, \ldots v_M\}$ of $(D^y)^{-1} L^y$ by solving the generalized eigenvalue problem $L^y v = \lambda D^y v$.

5 Set the number of clusters, k, as the number of eigenvalues below ρ or the lower bound.

6 Construct matrix $U = [v_1 \,|\, \cdots \,|\, v_k] \in \mathbb{R}^{|A^y| \times k}$.

7 Let u_a be the row of U that corresponds to a pattern a.

8 Use *k-means* to cluster the matrix rows $\{u_a\}_{a \in A^y}$ into clusters, $\{U_1, \ldots, U_k\}$.

output: clustering $P^y = \{C_1, C_2, \ldots, C_k\}$, where $C_i = \{a | u_a \in U_i\}$ for $i = 1, \ldots, k$

method [79], which requires only low-rank matrix-vector multiplications. The resulting algorithm scales linearly in the number of patterns to be clustered and can thereby be applied even to datasets with millions of patterns.

(iii) Cluster visualization and annotation. Our main assumption is that any cluster, $C \in P^y$, will correspond either to (part of) the object of interest, or to a distractor. To identify which of these possibilities it is, we introduce an efficient annotation step.

For each class, we randomly sample a small number, e.g. 12, of images from the training set. For each sampled image X we produce heatmaps, $H^y(X|C) \in \mathbb{R}^{|\mathcal{U}|}$, for each cluster, C, that depict for each region u the average similarity of the region pattern $\phi^y_u(X)$ to the patterns in the corresponding cluster, i.e.

$$H^y_u(X|C) = \frac{1}{|C|} \sum_{a \in C} \langle \phi^y_u(X), a \rangle. \tag{3.1}$$

Note that in practice we can compute this value without always summing over all patterns: we pre-compute the average cluster pattern, $a_C = \frac{1}{|C|} \sum_{a \in C} a$, and use $H^y_u(X|C) = \langle \phi^y_u(X), a_C \rangle$. We display the heatmaps and ask a human annotator to mark which of the clusters correspond to the object class of interest in the images, see Figure 3.2 for an illustration of the process. Overall, the annotation requires just one user interaction (a few mouse clicks) per class. Our experience shows that each interaction takes in the order of a few seconds, so a few hundred classes can be annotated within an hour.

(iv) Improved object class localization. The per-class annotations allow us to obtain better location predictions without having to retrain the network. Let X be a new image with predicted localization score map $S^y_u(X)$. For each location $u \in \mathcal{U}$ we determine the cluster that results

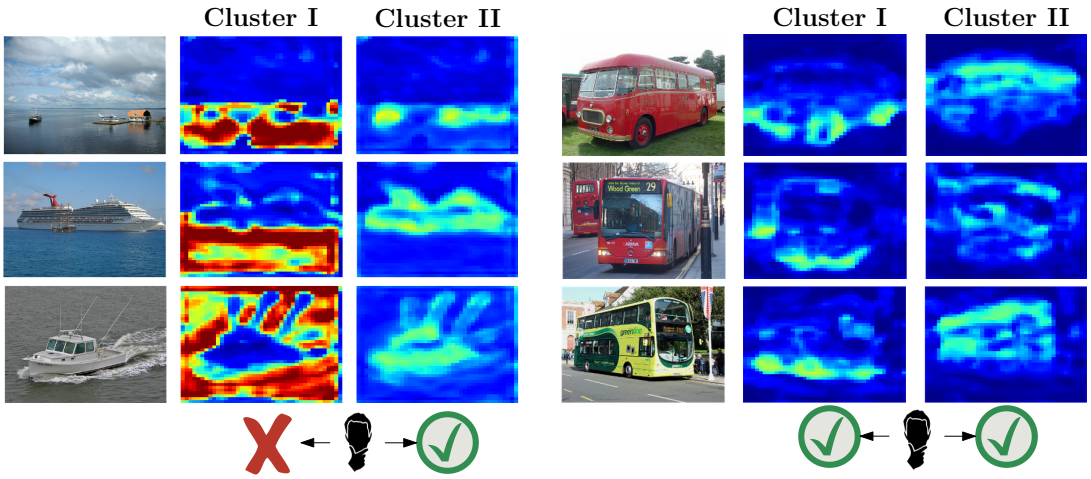


Figure 3.2: The schematic illustration of the annotation process. For any semantic category, we visualize corresponding mid-level feature clusters by heatmaps. An annotator marks every class as either representing an object of interest or background.

in the highest heatmap score, $C^* = \min_{C \in P^y} H_u^y(X|C)$. If C^* was not annotated as an 'object' cluster, we set $S_u^y(X)$ to $-\infty$, in order to prevent that the class y will be predicted at the location u.

3.4 Experiments

In this section we evaluate the effectiveness of our approach. We apply it to recently proposed methods for predicting object bounding boxes and semantic segmentations from image-level supervision and report experimental result on two challenging computer vision benchmarks: ILSVRC 2014 and PASCAL VOC 2012.

Predicting object bounding boxes. We follow the protocol of the ILSVRC 2014 *classification-with-localization* challenge: the goal is to predict which of 1000 object classes is present in an image and localize it by predicting a bounding box [122]. By the challenge protocol, up to five classes and their bounding boxes can be predicted, and the output is judged as correct if a bounding box of the correct class is predicted with an intersection-over-union score of at least 50% with a ground-truth box.

In this work we are particularly interested in the weakly-supervised setting, when models are trained using only per-image category information. We build on the GAP [171] technique, which uses a deep convolutional network with modified VGG [134] architecture. Internally, GAP produces localization score maps, $S^y(X)$, by means of the *CAM (class activation maps)* procedure, see [171] for details. GAP also includes a thresholding function: given a score map for a class y, all locations that have a with a score larger than 20% of the maximum score are

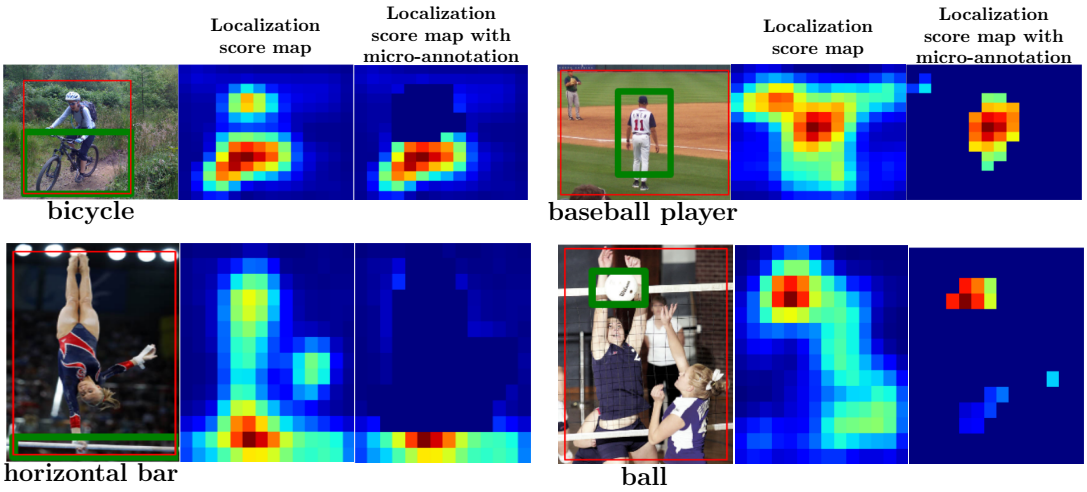


Figure 3.3: The effect of discarding localizations that correspond to a mid-level representation (pattern) that is assigned to a cluster annotated as distractor.

selected, i.e. $D^y(X) = \{u | S_u^y(X) > 0.2 \max_{u \in \mathcal{U}} S_u^y(X)\}$. At test time, for any input image five bounding boxes are produced, one for each class of the set of top-5 classes predicted by the convolutional neural network. For each predicted class the bounding box is produced, so that it covers the largest connected component of $D^y(X)$[1] Trained on the 1.2 million ILSVRC *train* images, this approach achieves an error rate of 49.9% on the ILSVRC *val* set.

Improving bounding box predictions. We use the proposed micro-annotation technique to improve the *CAM* localization score maps. For computing mid-level feature representations (patterns) we use the *conv5_3* layer of the modified VGG network from [171]. This choice is motivated by the closely related papers [172, 132] that studied object/part detectors emerging in convolutional neural networks. We set $\rho = 0.7$ as clustering threshold parameter and predict between 2 and 4 clusters. For the majority of classes (all except 56) we obtain only two clusters.

Annotating all clusters for the 1000 classes requires less then 6 hours of annotator time. In 182 semantic classes at least one cluster was identified as *distractor*, while for the remaining classes different clusters typically correspond to different object parts. See Figure 3.4 for a visualization of obtained object parts.

After modifying the localization scores according to our method we compute bounding boxes for all *val* images. By construction, only the localization scores for 182 classes are affected compared to the baseline GAP results. On average, the localization performance improved by 4.9% for those classes, with individual improvements up to 38% (*flatworm*). In Figure 3.5(a) we illustrate the results for the 79 classes for which localization score changed by at least 5% compared to the baseline. We observed that for vast majority of classes our method

[1] Better results can be achieved by using a more involved method based on multiple image crops. This was used for the results in [171], but is not described in the manuscript, so we use the simpler but reproducible setting.

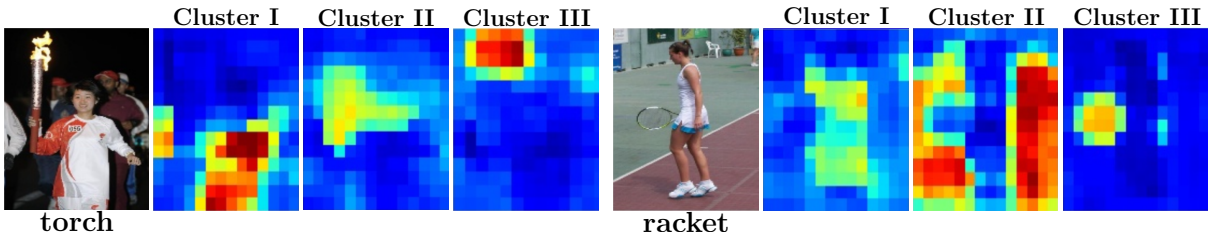


Figure 3.4: Examples of mid-level pattern clusters: *torch* has clusters that correspond to *person*, *torch* and *fire*, and *racket* has clusters that correspond to *player*, *court* and *racket*.

helps to improve localization performance. As expected, many of these are examples where object and background consistently co-occur, for example *speedboat* (improved from 42% to 80%) or *snowmobile* (improved from 32% to 60%). For a few classes, we observed a decrease of performance. We inspected these visually and observed a few possible reasons: one possibility is that distractors are present in the image, but they actually help to find a better bounding box: for example, drawing a bounding box around a complete person often achieves above 50% intersection-over-union for predicting *bath towels*. A second possibility is that objects consist of multiple visually disconnected parts, e.g. *sandals*. GAP's large-connected-component rule tends to fail for these, but by including distractors into the score map, such as the *foot*, can accidentally overcome this issue. We believe that this insight will be helpful for designing future weakly-supervised localization methods.

We additionally investigate the question whether the eigenvalues, which are produced by spectral clustering, can be leveraged to automatically identify classes with distractors. For this purpose we sort all 1000 classes by the second smallest eigenvalue given by spectral clustering in increasing order and study how the cumulative improvement of localization quality varies when micro-annotation is collected only for parts of the classes. Figure 3.5(b) depicts the curve, with the fraction of annotated classes on the x-axis and the fraction of localization improvement on the y-axis. One can see that the eigenvalues may be used to efficiently trade off annotation effort for localization performance. For example, it is sufficient to annotate 15% of all classes to obtain 50% of overall localization improvement, or 50% of the all classes for 85% of improvement.

Semantic image segmentation. Micro-annotation can also improve weakly-supervised semantic image segmentation. We follow the protocol of the *PASCAL VOC* challenges [42]: the goal is to produce segmentation masks by assigning one of 21 labels (20 semantic classes or background) to each pixel of an image. The evaluation metric is the mean intersection-over-union scores across all labels.

We are again interested in the weakly-supervised setting where models are trained only from per-image class annotation. Following the common practice, we use the *PASCAL VOC2012*

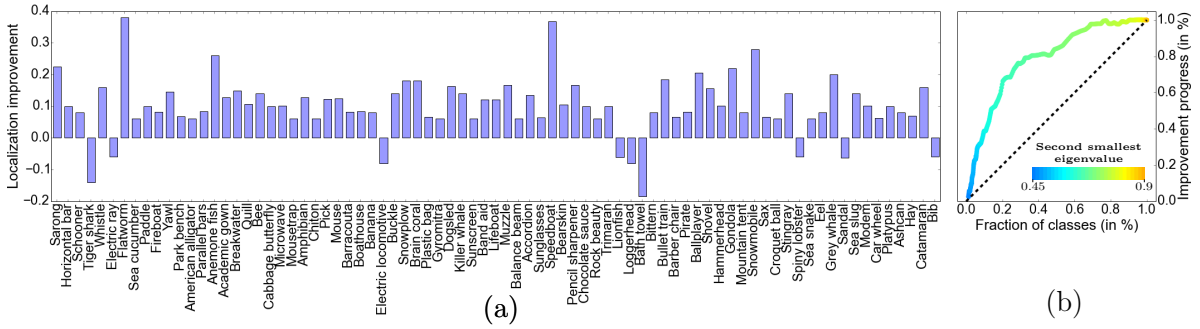


Figure 3.5: (a) Improvement of the localization scores by applying micro-annotation technique (for 79 classes with biggest changes); (b) Visualization of the trade off between the fraction of annotated classes and the fraction of overall improvement.

data with augmentation from [53]. In total, the training set (*train*) has 10,582 images. We report results on the validation part (*val*, 1449 images), and the test part (*test*, 1456 images). Because the ground truth annotations for the *test* set are not public, we rely on the independent evaluation server[2] to obtain numerical results for this data.

The technique for weakly-supervised image segmentation from image level labels is *SEC* [72]. Internally, it relies on *CAM* localization score maps, as introduced in the previous section, also based on a modified VGG-16 network [134] (but with different modifications), see [72] for details. In its original form, SEC achieves average intersection-over-union scores of 50.7% (val) and 51.7% (test).

Improving semantic image segmentation. We obtain mid-level pattern clusters for the 20 semantic classes of *PASCAL VOC* following the protocol as in the previous section. After annotating the clusters (which requires just a few minutes) we found two classes that have significant distractors: *boat* and *train*. Thus, we apply our method for these classes and retrain the *SEC* model using the improved localization score maps. The per-class numerical evaluation for the *val* and *test* sets is presented in Figure 3.7. We observe significant improvement of the intersection-over-union metric for the *boat* class (12.8% on *val*, 10.5% on *test*) and for the *train* class (14.2% on *val*, 17.7% on *test*). The performance for the other classes does not change significantly, only small perturbations occur due to the shared feature representation learned by the deep network. Overall, we achieve 52.2% and 53.0% mean intersection-over-union for the *val* and *test* sets, and improvement of 1.5% and 1.3% percent over the original *SEC* method. For a visual comparison of the segmentations predicted by the baseline and our approach see Figure 3.8.

[2] http://host.robots.ox.ac.uk:8080/

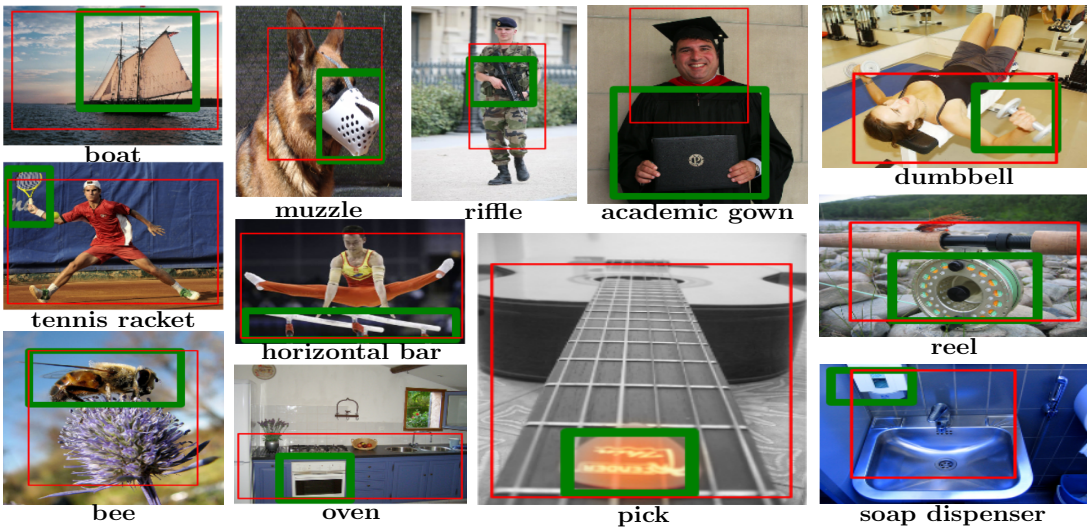


Figure 3.6: Typical mistakes (red boxes) of the baseline approach on ILSVRC *val* that are corrected by our method (green boxes). In particular, we demonstrate the following cases of foreground/background confusion: *boat/water*, *muzzle/dog*, *riffle/soldier*, *academic gown/academic hat*, *dumbbell/sportsman*, *tennis racket/tennis player*, *horizontal bar/gymnast*, *pick/guitar*, *reel/rod*, *bee/flower*, *oven/kitchen*, *soap dispenser/shell*.

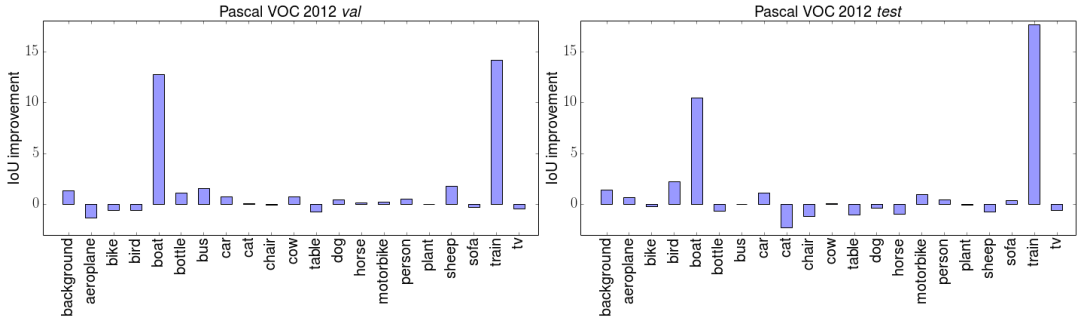


Figure 3.7: Improvement of the intersection-over-union scores by applying micro-annotation.

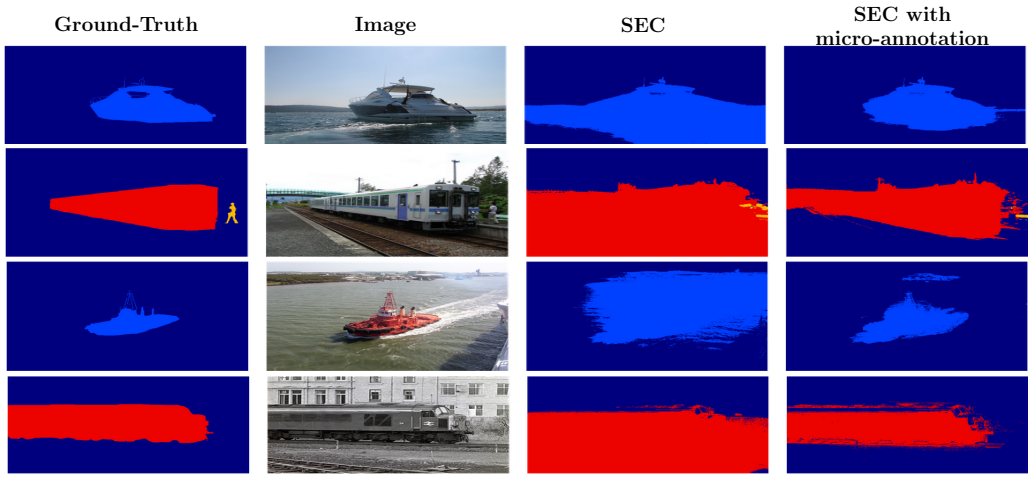


Figure 3.8: Examples of predicted segmentations masks without and with micro-annotation.

3.5 Conclusion

Weakly-supervised segmentation techniques have the inherent problem of confusing objects of interest with consistently co-occuring distractors. In this paper we present a micro-annotation technique that substantially alleviates this problem. Our key insight is that objects and distractors can be distinguished from each other because they form different clusters in the distributed representation learned by a deep network. We derive an annotation technique that requires only a few mouse clicks of user interaction per class and we propose an algorithm for incorporating the acquired annotation into the localization procedure.

Experiments on the *ILSVRC* 2014 and *PASCAL* 2012 demonstrate that the proposed micro-annotation method improves results for the competitive recently proposed weakly-supervised bounding box and semantic image segmentation prediction techniques.

4. Probabilistic Autoregressive Image Modeling

4.1 Introduction

In this chapter we present improved techniques for unsupervised probabilistic image modeling. Natural images are the main input to visual processing systems and, thus, understanding their structure is important for building strong and accurate automatic vision systems. Image modeling is also useful for a wide variety of key computer vision tasks, such as visual representation learning, automatic image colorization, inpainting, deblurring, super-resolution, image compression, and others.

Natural image modeling is known to be a very challenging statistical problem. Because the distribution over natural images is highly complex, developing models that are both accurate and computationally tractable is very challenging. Until recently, most of the existing models were restricted to modeling very small image patches, no bigger than, e.g., 9x9 pixels. Recently, however, deep convolutional neural networks (CNNs) have triggered noticeable advances in probabilistic image modeling. Out of these, PixelCNN-type models [141, 142, 124], have shown to deliver the best performance, while at the same time staying computationally tractable. However, PixelCNNs also have noticeable shortcomings: unless conditioned on external input, the samples they produce rarely reflect global structure of complex natural images, see [142, 124]. This raises concerns that current PixelCNN architectures might also be more limited than originally presumed. Moreover, PixelCNN's image sampling procedure is relatively slow in practice, as it requires to invoke a very deep neural network for every single image pixel that is to be generated.

In this work we derive improved PixelCNN models that address several of the aforementioned shortcomings. The main idea is to augment PixelCNN with appropriate auxiliary variables in order to isolate and overcome these drawbacks. This step, at the same time, provides us with important insights into the task of modeling natural images.

Besides the above insight, we make two main technical contributions in this chapter. First, we show that uncertainty in low-level image details, such as texture patterns, dominates the

objective of ordinary probabilistic PixelCNN models and, thus, these models may have little incentive to capture visually essential high-level image information, such as object shapes. We tackle this issue by deriving *Grayscale PixelCNNs* that effectively decouple the tasks of modeling low and high-level image details. This results in image samples of substantially improved visual quality. Second, we show that the sampling speed of PixelCNN models can be largely accelerated. We accomplish this by deriving *Pyramid PixelCNNs* that decompose the modeling of the image pixel probabilities into a series of much simpler steps. Employing a much lighter-weight PixelCNN architecture for each of them, globally coherent high-resolution samples can be obtained at reduced computational cost.

4.2 Related Work

Recent development of probabilistic image models based on deep neural networks, in particular variational auto-encoders (VAEs) and PixelCNNs sparked a rapid progress in unsupervised image modeling. In this chapter, we concentrate on the PixelCNN family of models [141, 142, 124] factorizes the distribution of a natural image using the elementary chain rule over pixels. The factors are modeled as deep convolutional neural networks with shared parameters and trained by maximum likelihood estimation.

VAEs [68] offer an alternative approach to probabilistic image modeling. They rely on a variational inequality to bound the intractable true likelihood of an image by a tractable approximation. VAEs are efficient to evaluate, but so far, produce results slightly worse than state-of-the-art PixelCNNs, both the likelihood scores and sampled images. Recent advances [50, 49, 5, 67, 52, 23] in VAEs literature exploit modifications of model structure, including usage of latent variables and autoregression principle, though these techniques remain technically and conceptually different from PixelCNNs.

Specifically for the task of producing images and other complex high-dimensional objects, *generative adversarial networks* (GANs) [46] have recently gained popularity. In contrast to PixelCNNs and VAEs, GANs are not explicit probabilistic models but feed-forward networks that are directly trained to produce naturally looking images from random inputs. A drawback of GAN models is that they have a generally unstable training procedure, associated with the search of a Nash equilibrium between two competing network players, and they can suffer from various technical problems, such as mode collapse or vanishing gradients. In order to make GANs work in practice, researchers resort to multiple non-trivial heuristics [123]. This strongly contrasts with probabilistic autoregressive models, such as PixelCNNs, which rely on well understood likelihood maximization for training and do not suffer from mode collapse problems.

Despite having fundamental differences on the technical level, PixelCNNs, VAEs and GANs may also benefit each other by sharing ideas. In particular, our work is related to the line of work on GANs with controllable image structure [113, 156]. A crucial difference of our work is, however, that our models do not require external supervision and, thus, remain purely unsupervised. Another notable paper in the context of this work introduces Laplacian GANs [33], with which we share the similar idea of using multi-scale decomposition for image generation. Similar constructions were suggested in [142] in the context of recurrent networks and [28] for the problem of super-resolution.

Very recent work [112], which appeared in parallel with ours, also addresses PixelCNN's sampling speed with multi-scale decomposition. Unlike our work, this paper makes strong additional independence assumption on the pixel level. We make a largely complementary contribution, as we explore different angles in which a multi-scale approach can improve performance.

4.3 PixelCNNs with Auxiliary Variables

In this section we remind the reader of the technical background and develop a framework for PixelCNNs with auxiliary variables. We then propose two new PixelCNN instances that provide insights into the natural image modeling task and lead to improved quality of sampled images and an accelerated sampling procedure. Finally, we conclude the section with implementation and training details.

We define an image $X = (x_1, x_2, \ldots, x_n)$ as a collection of n random variables associated with some unknown probability measure $p(X)$. Each random variable represents a 3-channel pixel value in the RGB format, where each channel takes a discrete value from the set $\{0, 1, 2, \ldots, 255\}$. The pixels are ordered according to a raster scan order: from left to right and from top to bottom. Given a dataset $\mathcal{D}$ of N images, our main goal is to estimate the unknown probability measure $p(X)$ from $\mathcal{D}$.

Recall that PixelCNNs are a family of models [141, 142, 124] that factorize the distribution of natural images using the basic chain rule:

$$p(X) = \prod_{j=1}^{n} p\left(x_j | x_1, \ldots, x_{j-1}\right). \tag{4.1}$$

Our key idea is to introduce an additional auxiliary variable, $\widehat{X}$, into the image modeling process. Formally, a PixelCNN with auxiliary variable is a probabilistic model of the joint distribution of X and $\widehat{X}$, factorized as

$$p(X, \widehat{X}) = p_{\hat{\theta}}(\widehat{X}) p_{\theta}(X | \widehat{X}), \tag{4.2}$$

where $[\theta, \hat{\theta}]$ are the model parameters. The conditional probability distribution, $p_\theta(X|\widehat{X})$, is modeled using the PixelCNN model [142], including the recent improvements suggested in [124]:

$$p_\theta(X|\widehat{X}) = \prod_{j=1}^{n} p_\theta\Big(x_j|x_1, \ldots, x_{j-1}; f_w(\widehat{X})\Big)\,, \tag{4.3}$$

where f_w is an embedding function parametrized by a parameter vector w.

Like other PixelCNN-based models, our model can be used for drawing samples. For a fixed $\hat{X}$, one follows the ordinary PixelCNN's sampling strategy. Otherwise, one first samples $\widehat{X}$ from $p_{\hat{\theta}}(\widehat{X})$ and then samples X from $p_\theta(X|\widehat{X})$ as described before.

Specifically, in this work we concentrate on auxiliary variables that 1) are a form of images themselves such that we can model $p_{\hat{\theta}}(\widehat{X})$ by a PixelCNN-type model, and 2) for which $\widehat{X}$ is approximately computable by a known deterministic function $\psi : X \to \widehat{X}$. This choice has the useful consequence that $p(\widehat{X}|X)$ is going to be a highly peaked distribution around the location $\psi(X)$, which provides us with a very efficient training procedure: denoting the model parameters $\Theta = (\hat{\theta}, \theta, w)$, we jointly maximize the log-likelihood of the observed training data $\mathcal{D}$ and the corresponding auxiliary variables, $\widehat{X} = \psi(X)$. More precisely, we solve

$$\begin{aligned}\Theta^* &= \operatorname*{argmax}_{\Theta} \sum_{X\in\mathcal{D}} \log p(X, \widehat{X}) \\ &= \operatorname*{argmax}_{\Theta} \sum_{X\in\mathcal{D}} \log p_\theta(X|\psi(X)) + \sum_{X\in\mathcal{D}} \log p_{\hat{\theta}}(\psi(X)).\end{aligned} \tag{4.4}$$

Because the objective function decomposes and the parameters do not interact, we can perform the optimization over $\hat{\theta}$ and (θ, w) separately, and potentially in parallel.

Note, that by this procedure we maximize a lower bound on the log-likelihood of the data:

$$\log \prod_{X\in\mathcal{D}} p(X, \widehat{X}) \leq \log \prod_{X\in\mathcal{D}} p(X) \tag{4.5}$$

By making the lower bound high we also guarantee that the log-likelihood of the data itself is high. Furthermore, we expect the bound (4.5) to be (almost) tight, as $p(X, \widehat{X}) = p(\widehat{X}|X)p(X)$, and by construction the first factor, $p(\widehat{X}|X)$ is (almost) a δ-peak centered at $\widehat{X} = \psi(X)$.

In the rest of this section we present two concrete realizations of the PixelCNNs augmented with auxiliary variables.

4.3.1 *Grayscale PixelCNN*

Despite great success of PixelCNN-type models, they are still far from producing plausible samples of complex natural scenes. A visual inspection of samples produced by the current state-of-the-art models reveals that they typically match low-level image details well, but fail at capturing global image structure, such as object shapes, see Figure 4.1 for an illustration.

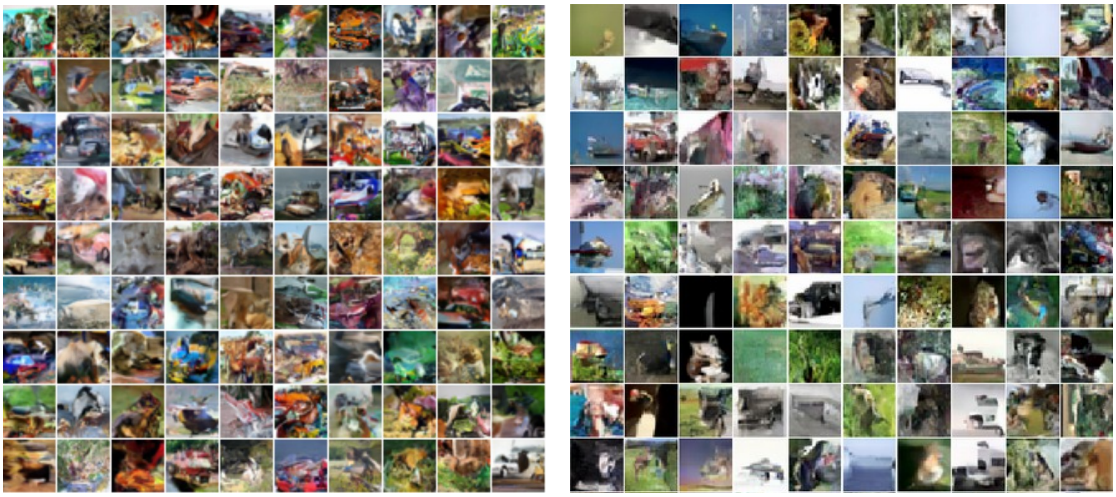


Figure 4.1: Samples produced by the current state-of-the-art autoregressive probabilistic image models: [142] *(left)* and [124] *(right)*.

We conjecture that a major reason for this is that the PixelCNN training objective provides too little incentive for the model to actually capture high-level image structure. Concretely, the PixelCNN's loss function (the negative data log-likelihood) measures the amount of uncertainty of the color value of each pixel, conditioned on the previous ones. This quantity is dominated by hard-to-predict low-level cues, such as texture patterns, which exhibit a large uncertainty. As a consequence, the probabilistic model is encouraged to represent such textures well, while visually more essential image details, such as object shapes, are neglected. We provide quantitative evidence for this claim in our experimental section. Similar findings are also discussed in [124].

In order to tackle the aforementioned shortcoming we derive *Grayscale PixelCNN*, in which the auxiliary variable $\widehat{X}$ is a 4-bit per pixel quantized grayscale version of the original 24-bits color image X. In combination with the factorization (4.2), this choice of auxiliary variable decouples the modeling of low and high-level image details: the distribution $p_{\hat{\theta}}(\widehat{X})$ (the quantized grayscale image) contains most information about global image properties, reflecting present objects and their shapes. The distribution $p_\theta(X|\widehat{X})$ models missing color and texture details. In Section 4.4 we highlight the quantitative and qualitative effects of augmenting PixelCNN with this choice of auxiliary variable.

4.3.2 *Pyramid PixelCNN*

In this section we address two further shortcomings of existing PixelCNN models. First, the strong asymmetry of the factors in Equation (4.1): the top left pixel of an image is modeled unconditionally, i.e. without any available information, while the bottom right pixel has access to the information of, potentially, all other pixel values. Nevertheless, the same network is

evaluated to model either of them (as well as all others pixels in between). We conjecture that it would be beneficial if pixels were generated with a less asymmetric usage of the image information.

Second, PixelCNNs have high computational cost for sampling images due to the recurrent nature of the procedure: for each generated pixel, the PixelCNN must invoke a convolutional neural network that is very deep, often in the order of a hundred convolutional layers. Note, that, in principle, at the sampling phase PixelCNN allows to cache intermediate values across consecutive PixelCNN invocations and, thus, save considerate computational effort. However, as reported in [110], caching delivers minor sampling speed gain, when only a few or a single image is generated at once, which is a common scenario in real-life applications.

In order to alleviate the aforementioned drawbacks we propose a PixelCNN model in which the auxiliary variable $\widehat{X}$ corresponds to a twice lower resolution view of X, thereby decoupling the full image model into a pair of simpler models: creating a lower resolution image, and upscaling a low-res into a high-res image. The upscaling step has strongly reduced model asymmetry, because all pixels on the high scale have equal access to all information from the lower scale. Also, by explicitly modeling the low-resolution image view we make it easier for the model to capture long-range image correlations, simply because the "long range" now stretches over fewer pixels.

Since the proposed auxiliary variable is an ordinary image, we can recursively apply the same decomposition to model the auxiliary variable itself. For example, if we apply decomposition recursively 4 times for modeling 128x128 images, it will result in a model which first generates an image on 8x8 resolution, and then upscales it 4 times by a factor 2. We call the resulting model *Pyramid PixelCNN*, as it resembles image pyramid decomposition.

Pyramid PixelCNNs break the image model into a series of simpler sub-models. Each sub-model is determined by the corresponding conditional distribution $p_\theta(X|\widehat{X})$ and the embedding $f_w(\widehat{X})$ (or just by $p_{\hat{\theta}}(\widehat{X})$ for the lowest resolution image). These conditional distributions model a relatively simple task of producing an image, given an embedding of the slightly lower resolution view of this image. Thus, we hypothesize that with an appropriate embedding function (potentially modeled by a very deep network), the conditional distributions can be reliably modeled using a very light-weight network. We then expect a significant sampling speed acceleration, because the major part of computational burden is redistributed to the embedding functions $f_w(\widehat{X})$, which needs to be computed only once per pyramid layer, not once for each pixel.

We quantify the modeling performance and sampling speed of the proposed multi-scale model later in Section 4.4.

4.3.3 Details of model parameterization

The PixelCNN model with auxiliary variable is fully defined by factors in (4.2), each of which is realized by a network with the PixelCNN++ architecture. This is described in details in [124]. The output of the PixelCNN++ is a 10-component mixture of three-dimensional logistic distributions that is followed by a discretized likelihood function [67, 124]. The only exception is the model for 4-bit quantized grayscale auxiliary variable $\widehat{X}$, where output is a vector of 16 probabilities for every possible grayscale value, followed by the standard cross-entropy loss.

The conditional PixelCNN model, $p(X|\widehat{X})$, depends on auxiliary variable $\widehat{X}$ in a fashion similar to [141, 124]. It can be summarized in two steps as follows: compute an embedding of $\widehat{X}$ using a convolutional network $f_w(\widehat{X})$, and bias the convolutions of every residual block by adding the computed embedding. We choose the architecture for modeling the embedding function, $f_w(\widehat{X})$, to be almost identical to the architecture of PixelCNN++. The main difference is that we use only one flow of residual blocks and do not shift the convolutional layers outputs, because there is no need to impose sequential dependency structure on the pixel level.

For numeric optimization, we use Adam [66], a variant of stochastic gradient optimization. During training we use dropout with 0.5 rate, in a way that suggested in [124]. No explicit regularization is used. Further implementation details, such as number of layers are specified later in Section 4.4.

4.4 Experiments

In this section we experimentally study the proposed *Grayscale PixelCNN* and *Pyramid PixelCNN* models on natural image modeling task and report quantitative and qualitative evaluation results.

4.4.1 *Grayscale PixelCNN*

Experimental setup. We evaluate the modeling performance of a *Grayscale PixelCNN* on the CIFAR-10 dataset [76]. It consists of 60,000 natural images of size 32×32 belonging to 10 categories. The dataset is split into two parts: a training set with 50,000 images and a test set with 10,000 images. We augment the training data by random horizontal image flipping.

For setting up the architectures for modeling the distributions $p_{\hat{\theta}}(\widehat{X})$, $p_\theta(X|\widehat{X})$ and embedding $f_w(X)$ we use the same hyperparameters as in [124]. The only exception is that for parameterizing $p_\theta(X|\widehat{X})$ and $f_w(\widehat{X})$ we use 24 residual blocks instead of 36.

In the Adam optimizer we use an initial learning rate of 0.001, a batch size of 64 images

Model	Bits per dim.
Deep Diffusion [135]	≤ 5.40
NICE [38]	4.48
DRAW [50]	≤ 4.13
Deep GMMs [143]	4.00
Conv Draw [49]	≤ 3.58
Real NVP [39]	3.49
Matnet + AR [5]	≤ 3.24
PixelCNN [142]	3.14
VAE with IAF [67]	≤ 3.11
Gated PixelCNN [141]	3.03
PixelRNN [142]	3.00
Grayscale PixelCNN	≤ 2.98
DenseNet VLAE [23]	≤ 2.95
PixelCNN++ [124]	2.92

Table 4.1: The negative log-likelihood of the different models for the CIFAR-10 **test set** measured as bits-per-dimension.

and an exponential learning rate decay of 0.99999 that is applied after each iteration. We train the grayscale model $p_{\hat{\theta}}(\widehat{X})$ for 30 epochs and the conditional model $p_{\theta}(X|\widehat{X})$ for 200 epochs.

Modeling performance. The *Grayscale PixelCNN* achieves an upper bound on the negative log-likelihood score of **2.98** bits-per-dimension. This is on par with current state-of-the art models, see Table 4.1. Note, that since we measure an upper bound, the actual model performance might be slightly better. However, in light of our and other experiments, we believe small differences in this score to be of minor importance, as the log-likelihood does not seem to correlate well with visual quality in this regime.

In Figure 4.2 we present random samples produced by the *Grayscale PixelCNN* model, demonstrating grayscale samples from $p_{\hat{\theta}}(\widehat{X})$ and resulting colored samples from $p_{\theta}(X|\widehat{X})$. We observe that the produced samples are highly diverse, and, unlike samples from previously proposed probabilistic autoregressive models, often exhibit a strongly coherent global structure, resembling highly complex objects, such as cars, dogs, horses, etc.

Given the high quality of the samples, one might be worried if possibly the grayscale model, $p_{\hat{\theta}}(\widehat{X})$, had overfit the training data. We observe that training and test loss of $p_{\hat{\theta}}(\widehat{X})$ are very close to each other, namely 0.442 and 0.459 of bits-per-dimension, which speaks against significant overfitting. Note also, that it is not clear if an overfitted model would automatically produce good samples. For instance, as reported in [124], severe overfitting of the PixelCNN++

Figure 4.2: Random quantized grayscale samples from $p(\widehat{X})$ *(top)* and corresponding image samples from $p(X|\widehat{X})$ *(bottom)*. The grayscale samples show several recognizable objects, which are subsequently also present in the color version.

Figure 4.3: CIFAR-10 images in original color *(left)* and quantized to 4-bit grayscale *(center)*. Images sampled from our conditional model $p(X|\widehat{X})$, using the grayscale CIFAR images as auxiliary variables *(right)*. The images produced by our model are visually as plausible as the original ones.

model does not lead to a high perceptual quality of sampled images.

Discussion. By explicitly emphasizing the modeling of high-level image structures in the *Grayscale PixelCNN*, we achieve significantly better visual quality of the produced samples. Additionally, the *Grayscale PixelCNN* offers interesting insights into the image modeling task. As the objective we minimize for training is a sum of two scores, we can individually examine the performance of auxiliary variable model $p(\widehat{X})$ and the conditional model $p(X|\widehat{X})$. The trained conditional model achieves $p(X|\widehat{X})$ a negative log-likelihood score of **2.52** bits-per-dimension, while $p(\widehat{X})$ achieves a score of **0.459** bits-per-dimension on the CIFAR-10 test set. In other words: high-level image properties, despite being harder to model for the network, contribute only a small fraction of the uncertainty score to the the overall log-likelihood. We take this as indication that if low-level and high-level image details are modeled by one model, then at training time low-level image uncertainty will dominate the training objective. We believe that this is this the key reason why previously proposed PixelCNN models failed to produce globally coherent samples of natural images. Importantly, this problem does not appear in the *Grayscale PixelCNN*, because global and local image models do not share parameters and, thus, do not interfere with each other at training phase.

An alternative explanation for the differences in log-likelihood scores would be that PixelCNN-type models are actually not very good at modeling low-level image input. Additional experiments that we performed show that this is not the case: we applied the learned conditional model $p(X|\widehat{X})$ to 4-bit grayscale images obtained by quantizing real images of the CIFAR-10 test set. Figure 4.3 compares the resulting colorized samples with the corresponding original images, showing that the samples produced by our conditional model are of visual quality comparable to the original images. This suggests that in order to produce even better image samples, mainly improved models for $p_{\hat{\theta}}(\widehat{X})$ are required.

4.4.2 *Pyramid PixelCNN*.

Experimental setup. We evaluate the *Pyramid PixelCNN* on the task of modeling face images. We rely on the *aligned&cropped CelebA* dataset [84] that contains approximately 200,000 images of size 218x178. In order to focus on human faces and not background, we preprocess all images in the dataset by applying a fixed 128x128 crop (left margin: 25 pixels, right margin: 25 pixel, top margin: 50, bottom margin: 40 pixels). We use a random 95% subset of all images as training set and the remaining images as a test set.

For the *Pyramid PixelCNN* we apply the auxiliary variable decomposition 4 times. This results in a sequences of probabilistic models, where the first model generates faces in 8×8 resolution. We use a PixelCNN++ architecture without down or up-sampling layers with only 3 residual blocks to model the distributions $p_\theta(\widehat{X})$ and $p_{\hat{\theta}}(X|\widehat{X})$ for all scales. For the embedding

Res.	8×8	16×16	32×32	64×64	128×128
Bpd	4.58	4.30	3.33	2.61	1.52

Table 4.2: Bits-per-dimension (*Bpd*) achieved by *Pyramid PixelCNN* on the test images from the *CelebA* dataset for various resolutions (*Res.*).

$f_w(\widehat{X})$ we use a PixelCNN++ architecture with 15 residual blocks with downsampling layer after the residual block number 3 and upsampling layers after the residual blocks number 9 and 12. For all convolutional layers we set the number of filters to 100.

In the Adam optimizer we use an initial learning rate 0.001, a batch size of 16 and a learning rate decay of 0.999995. We train the model for 60 epochs.

Modeling performance. We present a quantitative evaluation of *Pyramid PixelCNN* in Table 4.2. We evaluate the performance of our model on different output resolutions, observing that bits-per-dimension score is smaller for higher resolutions, as pixel values are more correlated and, thus, easier to predict. As an additional check, we also train and evaluate *Pyramid PixelCNN* model on the CIFAR dataset, achieving a competitive score of 3.32 bits-per-dimension.

Before demonstrating and discussing face samples produced by our model we make an observation regarding the PixelCNN's sampling procedure. Recall that the output of the *Pyramid PixelCNN* is a mixture of logistic distributions. We observe an intriguing effect related to the mixture representation of the predicted pixel distributions for face images: the perceptual quality of sampled faces substantially increases if we artificially reduce the predicted variance of the mixture components. We illustrate this effect in Figure 4.4, where we alter the variance by subtracting constants from a fixed set of $\{0.0, 0.1, \ldots, 1.0\}$ from the predicted log-variance of the mixture components.

Inspired by this observation we propose an alternative sampling procedure: for each pixel, we randomly sample one of the logistic components based on their weight in the predicted mixture. Then, we use the mode of this component as sampled pixel value, instead of performing a second random sampling step. This sampling procedure can be seen as a hybrid of probabilistic sampling and maximum a posteriori (MAP) prediction.

Figure 4.5 shows further samples obtained by such *MAP sampling*. The produced images have very high perceptual quality, with some generated faces appearing almost photo-realistic. The complete multi-scale sampling mechanism of the *Pyramid PixelCNN*, from 8×8 to 128×128 images, is demonstrated in Figure 4.6.

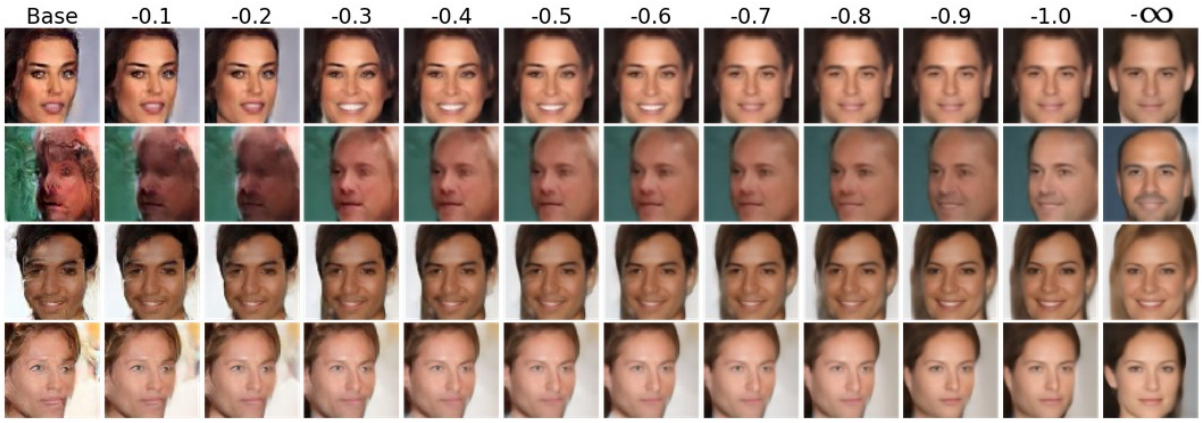


Figure 4.4: Effect of the variance reduction. Numbers on top of each column indicates the amount of reduction in the predicted log-variance of the mixture components. The last column corresponds to MAP sampling.

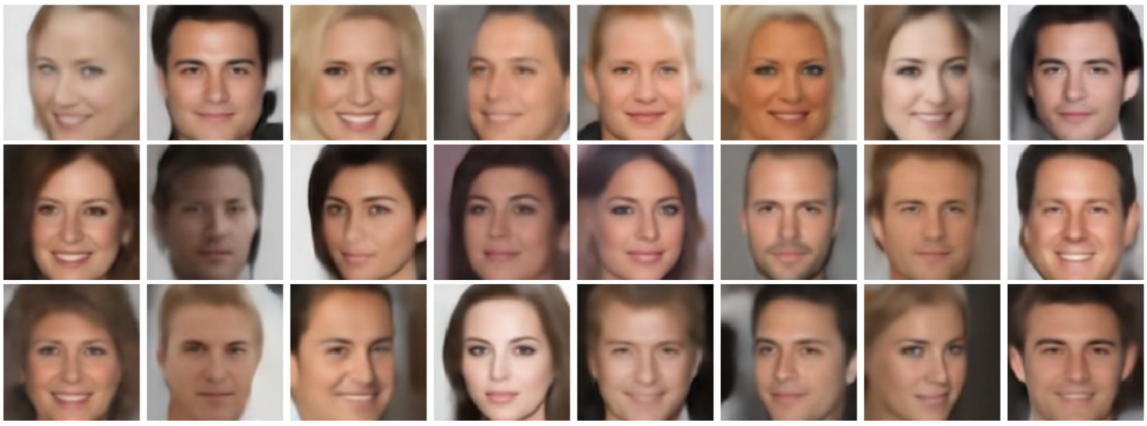


Figure 4.5: Images sampled from the *Pyramid PixelCNN* by MAP sampling. The generated faces are of very high quality, many being close to photorealistic. At the same time, the set of sample is diverse in terms of the depicted gender, skin color and head pose.

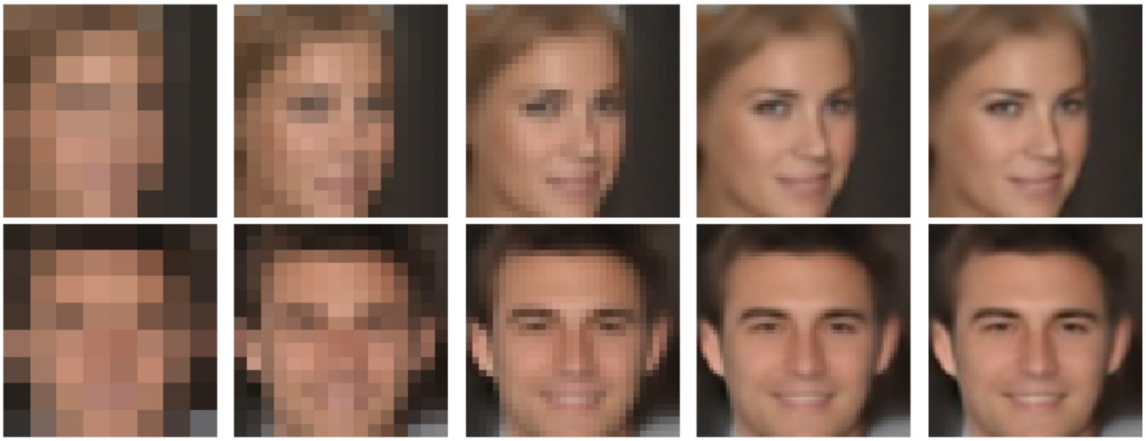


Figure 4.6: Visualization of the *Pyramid PixelCNN* sampling process. Faces are enerated on a small, 8x8, resolution and then are upsampled until reaching the desired 128x128 resolution.

Discussion. First, we observe that, despite the very high resolution of modeled images, the produced samples capture global human face characteristics, such as arrangement of face elements and global symmetries. At the same time, the set of samples is diverse, containing male as well as female faces, different hair and skin colors as well as facial expressions and head poses. Second, we emphasize that by properly decomposing the model we are able to scale the *Pyramid PixelCNN* to produce samples with very high resolution of 128x128. As discussed previously in Section 4.3, this results from the fact that our decomposition allows to parametrize autoregressive parts of the image model by a light-weight architecture. Concretely, on an NVidia TitanX GPU, our *Pyramid PixelCNN* without caching optimizations requires approximately **0.004** seconds on average to generate one image pixel, while a PixelCNN++ even with recently suggested caching optimizations requires roughly **0.05** seconds for the same task. If we add caching optimizations to our model, we expect its speed to improve even further.

4.5 Conclusion

In this paper we presented *Grayscale PixelCNN* and *Pyramid PixelCNN*, an improved autoregressive probabilistic techniques that incorporate auxiliary variables. We derived two generative image models that exploit different image views as auxiliary variables and address known limitations of existing PixelCNN models. The use of quantized grayscale images as auxiliary variables resulted in a model that captures global structure of complex natural images and produces globally coherent samples. With multi-scale image views as auxiliary variable, the model was able to efficiently produce realistic high-resolution images of human faces. Note, that these improvements are complementary and we plan to combine them in a future work.

Furthermore, we gained interesting insights into the image modeling problem. First, our experiments suggest that texture and other low-level image information distract probabilistic models from focusing on more essential high-level image information, such as object shapes. Thus, it is beneficial to decouple the modeling of low and high-level image details. Second, we demonstrate that multi-scale image model, even with very shallow PixelCNN architectures, can accurately model high-resolution face images.

5. Automatic Image Colorization

5.1 Introduction

In this chapter we use general-purpose density estimation techniques from the previous chapter and derive the state-of-the-art probabilistic image colorization method.

Previously proposed colorization models are able to capture the evident mappings abounding in the training data, e.g., blue sky, but often lack two main appealing properties: (i) *diversity*, i.e. being able to produce several plausible colorizations, as there is generally no unique solution, and (ii) *color vibrancy* of the produced samples; the colorized images should display proper level of saturation and contrast like natural images, not look desaturated.

Most state-of-the-art colorization techniques do not in fact offer a proper sampling framework in the sense that they only model pixelwise color distributions rather than a joint distribution for colors of natural images. In contrast, our model relies on recent advances in autoregressive PixelCNN-type networks [141, 67] for image modeling. Specifically, our architecture is composed of two networks. A deep feed-forward network maps the input grayscale image to an embedding, which encodes color information, much like current state-of-the-art colorization schemes. This embedding is fed to an autoregressive network, which predicts a proper distribution of the image chromaticity conditioned on the grayscale input. Modeling the full multimodal joint distribution over color values offers a solution to the *diversity* problem, as it provides us with a simple, computationally efficient, and yet powerful probabilistic framework for generating different plausible colorizations. Furthermore, the model likelihood can be used as a principled quantitative evaluation measure to assess the model performance.

As we discuss in this chapter, the problem of color vibrancy is a consequence of not modeling pixel interactions and is hard to tackle in a principled way. In particular, [168] addresses it by (i) treating colorization as a classification task, avoiding the problem of using a regression objective which leads to unimodal, and thus, desaturated predictions, and (ii) introducing rebalancing weights to favor rare colors present in natural images and more difficult to predict. In the experiments section, we show that our model generally produces vivid samples, without any *ad hoc* modifications of the training procedure.

In Section 5.3 we introduce the theoretical framework to support the autoregressive component of our model, as well as our training and inference procedures. We report experimental results in Section 5.4, including qualitative comparison to several recent baselines.

5.2 Related work

Automatic image colorization has been a goal of Image Processing and Computer Vision research since at least the 1980s, after movie studies started releasing re-colorized movies from the black-and-white era [92]. Because manually colorizing every frame of a movie is very tedious and expensive work, semi-automatic systems soon emerged, e.g. based on the manual colorization of key frames followed by motion-based color propagation [88]. Subsequently, techniques that required less and less human interaction were developed, e.g., requiring only user scribbles [82, 64], reference color images [19, 89], or scene labels [36].

Succesful fully automatic approaches emerged only recently [58, 80, 168, 60, 35, 15] based on deep architectures. A straight-forward approach is to train a convolutional feedforward model to independently predict a color value for each pixel [58, 80, 168]. However, these techniques do not model crucial interactions between pixel colors of natural images, and thus, probabilistic sampling yields high-frequency patterns of low perceptual quality (see Figure 5.1). Predicting the mode or expectation of the learned distribution instead results in grayish, and still often noisy colorizations (see, e.g., Figure 5.6). Recent unpublished work [60] proposes to train colorization model using generative adversarial networks (GANs) [46]. GANs, however, are known to suffer from unstable training and lack of a consistent objective, which often prevents a quantitative comparison of models.

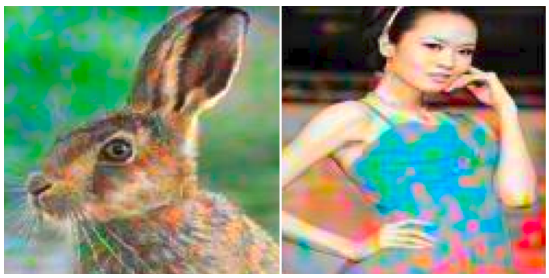


Figure 5.1: Colorized samples from a feedforward model.

A shared limitation of the models discussed above is their lack of *diversity*. They can only produce one colored version from each grayscale image, despite the fact that are typically multiple plausible colorizations. [15] for instance addresses the problem in the framework of conditional GANs. To our knowledge, the only work besides ours aiming at representing a fully probabilistic multi-modal joint distribution of pixel colors is [35]. It relies on the variational autoencoder framework [68], which, however, tends to produce more blurry outputs than other image generating techniques. In contrast, the autoregressive [142, 124] network that we employ is able to produce crisp high-quality and diverse colorizations.

Concurrently to this thesis, Guadarrama et al. [51] proposed a similar automatic image colorization technique, which is also based on the autoregressive networks.

5.3 Probabilistic Image Colorization

In this section we present our **P**robabilistic **I**mage **C**olorization model (PIC). We first introduce the technical background, then formulate the proposed probabilistic model and conclude with parametrization and optimization details.

5.3.1 Background

Let X be a natural image containing n pixels, indexed in raster scan order: from top to bottom and from left to right; the value of the i-th pixel is denoted as X_i. We assume that images are encoded in the LAB color space, which has three channels: the luminance channel (**L**) and the two chrominance channels (**a** and **b**). We denote by X^L and X^{ab} the projection of X to its luminance channel and chrominance channels respectively. By convention a **Lab** triplet belongs to the range $[0; 100] \times [-127; 128] \times [-128; 127]$. Consequently, each pixel in X^{ab} can take $256 \times 256 = 65536$ possible values.

Our goal is to predict a probabilistic distribution of image colors from an input gray image (luminance channel), i.e. we model the conditional distribution $p(X^{ab}|X^L)$ from a set of training images, $\mathcal{D}$. This is a challenging task, as X^{ab} is a high dimensional object with a rich internal structure.

5.3.2 Modeling the joint distribution of image colors

To tackle the aforementioned task we rely on recent advances in autoregressive probabilistic models [142, 124]. The main insight is to use the chain rule in order to decompose the distribution of interest into elementary per-pixel conditional distributions; all of these distributions are modeled using a shared deep convolutional neural network:

$$p(X^{ab}|X^L) = \prod_{i=1}^{n} p(X_i^{ab}|X_1^{ab}, \ldots, X_{i-1}^{ab};\ X^L). \tag{5.1}$$

Note, that (5.1) makes no assumptions on the modeled distribution. It is only an application of the chain rule of probability theory. At training time, all variables in the factors are observed, so a model can be efficiently trained by learning all factors in parallel. At test time, we can draw a sample from the joint distribution using a pixel-level sequential procedure: we first sample X_1^{ab} from $p(X_1^{ab}|X^L)$, then sample X_i^{ab} from $p(X_i^{ab}|X_1^{ab}, \ldots X_{i-1}^{ab}; X^L)$ for all i in $\{2 \ldots n\}$.

We denote the deep autoregressive neural network for modeling factors from (5.1) as f^θ, where θ is a vector of parameters. The autoregressive network f^θ outputs a vector of normalized probabilities over the set, $\mathcal{C}$, of all possible chrominance **(a, b)** pairs. For brevity, we denote a predicted probability for the pixel value X_i^{ab} as f_i^θ. To model the dependency on the observed grayscale image view X^L we additionally introduce a deep neural network $g^w(X^L)$, which produces a suitable embedding of X^L. To summarize, formally, each factor in (5.1) has the following functional form:

$$p(X_i^{ab}|X_1^{ab},\ldots,X_{i-1}^{ab};\ X^L) = f_i^\theta(X_1^{ab},\ldots,X_{i-1}^{ab};\ g^w(X^L)) \tag{5.2}$$

Note, that the autoregressive network f^θ outputs a probability distribution over all color values in $\mathcal{C}$. The standard way to encode such a distribution over discrete values is to parametrize f^θ to output a score for each of the possible color values in $\mathcal{C}$ and then apply the softmax operation to obtain a normalized distribution. In our case, however, the output space is huge (65536 values per pixel), and the standard approach has crucial shortcomings: it will result in a very slow convergence of the training procedure and will require a vast amount of data to generalize. It is possible to alleviate this shortcoming by quantizing the colorspace at the expense of a slight drop in colorization accuracy and possible visible quantization artifacts. Furthermore, it still results in a large number of classes, typically a few hundreds, leading to slow convergence; additional heuristics, such as soft label encoding [168], are then required to speed up the training.

Instead, we approximate the distribution in (5.2) with a mixture of 10 logistic distributions, as described in [124]. This requires f^θ to output the mixture weights as well as the first and second-order statistics of each mixture. In practice, we need less than 100 output values per pixel to encode those, which is significantly fewer than for the standard discrete distribution representation. This model is powerful enough to represent a multimodal discrete distribution over all values in $\mathcal{C}$. Furthermore, since the representation is partially continuous, it can make use of the distance of the color values in the real space, resulting in faster convergence.

In the rest of the section section we give details on the architecture for g^w and f^θ and on the optimization procedure.

5.3.3 Model architecture and training procedure

We present a high-level overview of our model in Figure 5.2. It has two major components: the embedding network g^w and the autoregressive network f^θ. Intuitively, we expect that g^w, which only has access to the grayscale input, produces an embedding encoding information about plausible image colors based on the semantics available in the grayscale image. The

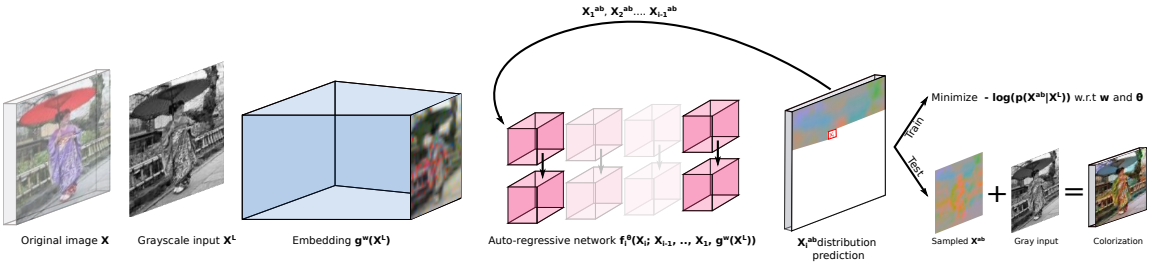


Figure 5.2: High-level model architecture for the proposed model

autoregressive network then makes use of this embedding to produce the final colorization, while being able to model complex interactions between image pixels.

Our design choices for parametrizing networks g^w and f^θ are motivated by [124], as it reports state-of-the-art results for the challenging and related problem of natural image modeling. In particular, we use *gated residual blocks* as the main building component for the both networks. Each residual block has 2 convolutions with 3x3 kernels, a skip connection [54] and gating mechanism [141, 124]. Convolutions are preceded by concatenated [128] exponential linear units [26] as non-linearities and parametrized as proposed in [125]. If specified, the first convolution of the residual block may have a dilated receptive field [164]; we use dilation to increase the network's field-of-view without reducing its spatial resolution.

The embedding network g^w is a standard feed-forward deep convolutional neural network. It consist of gated residual blocks and (strided) convolutions. We give more precise details on the architecture in the experimental section.

For parametrizing f^θ we use the *PixelCNN++* architecture from [124]. On a high level, the network consists of two flows of residual blocks, where the output of every convolution is properly shifted to achieve sequential dependency: X_i^{ab} depends only on $X_1^{ab}, \ldots, X_{i-1}^{ab}$. Conditioning on the external input, X^L, is achieved by biasing the output of the first convolution of every residual block by the embedding $g^w(X^L)$. We use no down- or up-sampling layers. For more detailed explanation of this architecture see our implementation or [124].

Spatial chromatic subsampling. It is known that the human visual system resolves color less precisely than luminance information [144]. We exploit this fact by modeling the chrominance channels at a lower resolution than the input luminance. This allows us to reduce computational and memory requirements without losing perceptual quality. Note that image compression schemes such as JPEG or previously proposed techniques for automatic colorization also make use of chromatic subsampling.

CIFAR-10 embedding $g^w(X^L)$			
Operation	Res.	Width	D
Conv. 3x3/1	32	32	–
Resid. block $\times$ 2	32	32	–
Conv. 3x3/2	16	64	–
Resid. block $\times$ 2	16	64	–
Conv. 3x3/1	16	128	–
Resid. block $\times$ 2	16	128	–
Conv. 3x3/1	16	256	–
Resid. block $\times$ 3	16	256	2
Conv. 3x3/1	16	256	–

ILSVRC 2012 embedding $g^w(X^L)$			
Operation	Res.	Width	D
Conv. 3x3/1	128	64	–
Resid. block $\times$ 2	128	64	–
Conv. 3x3/2	64	128	–
Resid. block $\times$ 2	64	128	–
Conv. 3x3/2	32	256	–
Resid. block $\times$ 2	32	256	–
Conv. 3x3/1	32	512	–
Resid. block $\times$ 3	32	512	2
Conv. 3x3/1	32	512	–
Resid. block $\times$ 3	32	512	4
Conv. 3x3/1	32	512	–

Table 5.1: Architecture of g^w for the CIFAR-10 and ILSVRC 2012 datasets. The notation "$\times$ k" in the **Operation** column means the corresponding operation is repeated k times. **Res.** is the layer's spatial resolution, **Width** is the number of channels and **D** is the dilation rate.

Optimization. We train the parameters θ and w by minimizing the negative log-likelihood of the chrominance channels in the training data:

$$\arg\min_{\theta,w} \sum_{X \in \mathcal{D}} -\log p(X^{ab}|X^L) \tag{5.3}$$

We use the Adam optimizer [66] with an initial learning rate of 0.001, momentum of 0.95 and second momentum of 0.9995. We also apply Polyak parameter averaging [106].

5.4 Experiments

In this section we present quantitative and qualitative evaluation of the proposed probabilistic image colorization (PIC) technique. We evaluate our model on two challenging image datasets: CIFAR-10 and ImageNet ILSVRC 2012. We also qualitatively compare our method to previously proposed colorization approaches and perform additional studies to better understand various components of our model. Our Tensorflow implementation and pre-trained models are publicly available[1].

[1] Probabilistic Image Colorization, https://github.com/ameroyer/PIC

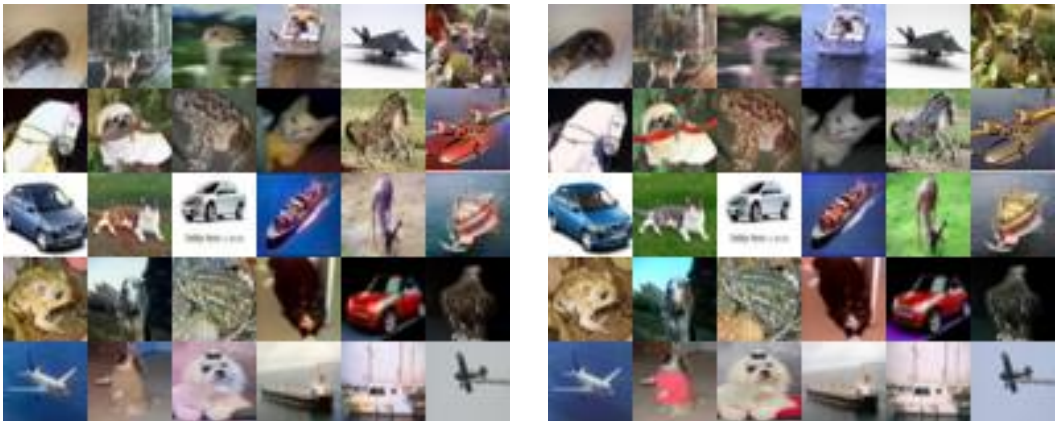


Figure 5.3: Colorized image samples from our model (left) and the corresponding original CIFAR-10 images (right). Images are selected randomly from the test set.

5.4.1 CIFAR-10 experiments

We first study the colorization abilities of our method on the CIFAR-10 dataset, which contains 50000 training images and 10000 test images of 32x32 pixels, categorized in 10 semantic classes. We fix the architecture of the embedding network g^w as specified in Table 5.1 (left). For the autoregressive network f^θ we use 4 residual blocks and 160 output channels for every convolution. We subsample the spatial chromatic resolution by a factor of 2, i.e. model the color channels on the resolution of 16x16. We train the resulting model as explained in Section 5.3 with batch size of 64 images for 150 epochs. The learning rate decays after every training iteration with constant multiplicative rate 0.99995.

In Figure 5.3 we visualize random test images colorized by PIC (left) and the corresponding real CIFAR-10 color images (right). We note that the samples produced by PIC appear to have natural colors and are hardly distinguishable from the real ones. This speaks in favour of our model being appropriate for modeling the color distribution of natural images.

We also report that PIC achieves a negative log-likelihood of 2.72, measured in bits-per-dimension. Intuitively, this measure indicates the average amount of uncertainty in the image colors under the trained model. This is a principled measure that can be used to perform model selection and compare various probabilistic colorization techniques.

5.4.2 ILSVRC 2012 experiments

After successful preliminary experiments on the CIFAR-10 dataset we now present experimental evaluation of PIC on the much more challenging ILSVRC 2012. This dataset has 1.2 million high-resolution training images spread over 1000 different semantic categories, and a hold-out set of 50000 validation images. In our experiments we rescale all images to 128x128 pixels,

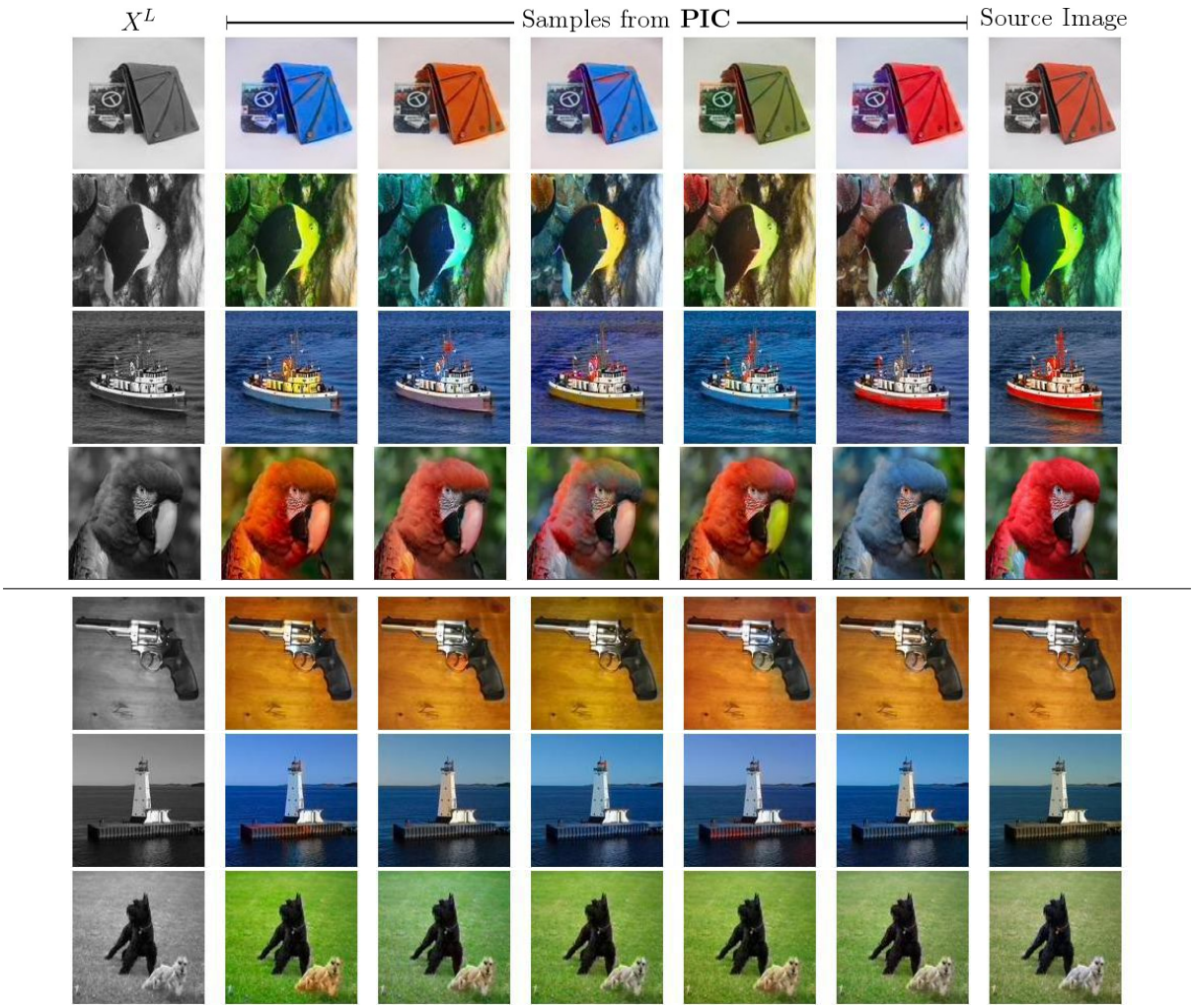


Figure 5.4: Colorized samples from our model illustrate its ability to produce diverse (top) or consistent (bottom) samples depending whether the image semantics are ambiguous or not.

which is enough to capture essential image details and remain a challenging scenario.

As ILSVRC images are of higher resolution and contain more details than CIFAR-10 images, we use a slightly bigger architecture for the embedding function g^w as specified in Table 5.1 (right) and a chroma subsampling factor of 4, as in [168]. The autoregressive component f^θ has 4 residual blocks and 160 channels for every convolution.

We run the optimization algorithm for 20 epochs using batches of 64 images, with learning rate decaying multiplicatively after every iteration with the constant of 0.99999.

In Figure 5.4 we present successfully colorized images from the validation set. These demonstrate that our model is capable of producing spatially coherent and semantically plausible colors. Moreover, as expected, in the case where the color is ambiguous, the produced samples often demonstrate wide color diversity. Nevertheless, if the color is unambiguously determined by the semantics of the object (grass or sky), then PIC produces consistent colors.

To provide further insight, we also highlight two failure cases in Figure 5.5: First, PIC may

Figure 5.5: Illustration of failure cases: PIC may fail to reflect very long-range pixel interactions (top) and, e.g., assign different colors to disconnected parts of an occluded object, or it may fail to understand semantics of complex scenes with unusual objects (bottom).

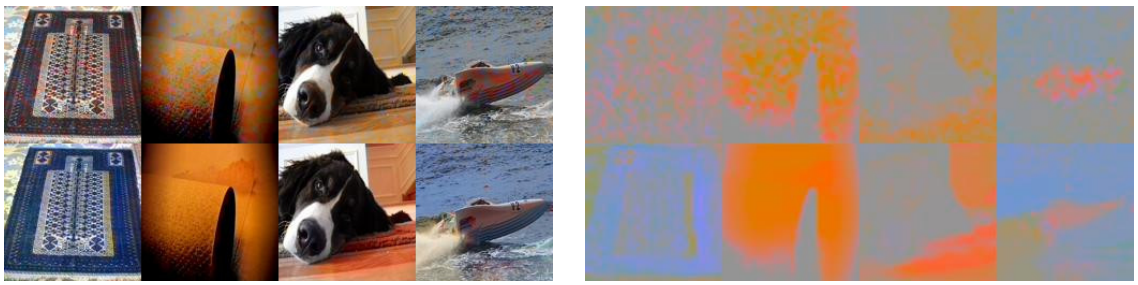


Figure 5.6: Comparison on ImageNet validation set between MAP samples from the embedding network g_w *(top)* and random samples from the autoregressive PIC model *(bottom)*. Colored image (left) and predicted chrominances for fixed $\mathbf{L} = 50$ (right)

not fully capture complex long-range pixel interactions interactions, e.g., if an object is split due to occlusion, the two parts may have different colors. Second, for some complex images with unusual objects PIC may fail to understand semantics of the image and produce not visually plausible colors.

Our model achieves a negative log-likelihood of 2.51 bits-per-dimension. Note that purely generative model from [142], which is based on the similar but deeper architecture, reports a negative log-likehood of 3.86 for the ILSVRC validation images modeled on the same resolution. As our model has access to additional information (grayscale input), it is not surprising that we achieve better likelihood; Nevertheless, this result confirms that PIC learns non-trivial colorization model and strengthens our qualitative evaluation.

5.4.3 Importance of the autoregressive component

One of the main novelties of our model is the autoregressive component, f^θ, which drastically increases the colorization performance by modeling the joint distribution over all pixels. In this section we perform an ablation study in order to investigate the importance of the autoregressive component alone. Note that without f^θ, our model essentially becomes a standard

Figure 5.7: Qualitative results from several recent automatic colorization methods compared to the original (right) and sample from our method (first to last column).

feed-forward neural network, similar to recent colorization techniques [168, 80], Specifically, we use PIC pretrained on the ILSVRC dataset, discard the autoregressive component f^{θ}, and finetune the remaining embedding network, g^{w}, for the task of image colorization. At test time, we use maximum a posteriori (MAP) sampling from this model. Stochastic sampling from the output of g^{w} would produce very noisy colorizations as the pixelwise predicted distributions are independent. Alternatively, one could predict the mean color of the predicted distribution for each pixel, but that would produce mostly gray colors.

From comparing the output samples of PIC and g^{w}, it appears the benefit brought by the autoregressive component is two-fold: first, it explicitly models relationships between neighboring pixels, which leads to visually smoother samples as can be seen in Figure 5.6. Second, the samples generated from PIC tend to display more saturated colors. This is due to the fact that our model allows for proper probabilistic sampling and, thus, can produce rare and globally consistent colors. We also verify that PIC produces more vivid colors by computing the average perceptual saturation [87]. Based on 1000 random image samples, the PIC model and g^{w} have an average saturation of 36.4% and 32.7%, respectively.

5.4.4 Qualitative comparison to baselines.

In Figure 5.7 we present a few colorization results on the ImageNet validation set for our model (one sample) as well as three recent colorization baselines. **Zhang et al., 2016** [168] proposes a deep VGG architecture trained on ImageNet for automatic colorization. The main innovation is that they treat colorization as a classification rather than regression task, combined with class-rebalancing in the training loss to favor rare colors and more vibrant samples. **Larsson et al., 2016** [80] is very similar to the first baseline, except for a few architectural differences (e.g., use of hypercolumns) and heuristics. **Iizuka et al., 2016** [58] proposes a non-probabilistic model with a regression objective. Their architecture is also more complex as they use two distinct flows for local and global features. We also note that their model was trained on the MIT Places dataset, while ours and the two previous baselines use ImageNet. We use the publicly available implementation for each baseline.

In general, we observe that our model is highly competitive with other approaches and tends to produce more saturated colors on average.

5.5 Conclusion

Deep feedforward networks achieve promising results on the task of colorizing natural gray images. The generated samples however often suffer of a lack of *diversity* and *color vibrancy*. We tackle both aspects by modelling the full joint distribution of pixel color values using an autoregressive network conditioned on a learned embedding of the grayscale image. The fully probabilistic nature of this framework provides us with a proper and straightforward sampling mechanism, hence the ability to generate diverse samples from a given grayscale input. Furthermore, the data likelihood can be efficiently computed from the model and used as a quantitative evaluation metric. We report quantitative and qualitative evaluations of our model, and show that colorizations sampled from our architecture often display vivid colors, indicating that the model captures well the underlying color distribution of natural images, without requiring any *ad hoc* heuristics during training.

6. Conclusion and Future Work

Weakly-supervised image segmentation. The first part of the thesis is dedicated to the semantic image segmentation task. In this part we proposed a new model for weakly-supervised image segmentation, which can learn from only image-level labels and outperforms previously proposed techniques under the same experimental conditions. We also identified inherent failure mode of all weakly-supervised models and derived a novel micro-annotation technique, which helps to mitigate the identified problem. In particular, we observed that weak supervision can not resolve some ambiguities present in the data and developed a methodology for gathering tiny amount of additional supervision to correctly resolve these ambiguities. The proposed weakly-supervised model, in conjunction with the micro-annotation technique, significantly reduces the gap between performance of weakly-supervised and supervised image segmentation models.

Nevertheless, despite substantial progress in improving performance of weakly-supervised models, there is still a large room for improvement. Consider, for instance, human visual systems, which are typically very good at producing segmentation masks, despite lack of fully supervised training for solving this particular task. In order to identify potential directions for further improvements we first summarize key technical observations from Chapter 2.

Evidently, image-level labels alone do not provide enough information to meaningfully solve the image segmentation task. As a result, currently successful weakly-supervised techniques, including ours, explicitly or implicitly utilize additional prior knowledge about the nature of the image segmentation task. Specifically, in a weakly-supervised model, which is proposed in this thesis, we utilize the following prior assumptions on:

1. *Locality of semantic categories in natural images*, i.e. if a certain semantic class appears in an image, then it can be recognized by analyzing a certain local image region.

2. *Characteristic sizes*, where we presume that object's pixels typically occupy a certain fraction of all image pixels, which is bounded away from 0 or 1.

3. *Smoothness* of segmentation masks, which means that neighboring pixels are likely to have the same semantic label.

We use these assumptions to derive a specialized loss function for learning segmentation models from image-level labels.

We hypothesize that further improvements are possible through careful revision or augmentation of these assumptions. In particular, we believe that it is promising to explore prior knowledge, which can be learned from external data. For instance, some of the earlier work already explores such priors, e.g. *objectness prior* [149, 8] or prior learned from motion cues [101, 138]. Perhaps, an important piece, which is missing now, is learning from text-based information. Rich text-based sources may reveal hierarchy and relations between different concepts and strongly boost performance of weakly-supervised image segmentation models.

Another promising direction for improving weakly-supervised image segmentation is development of intelligent annotation systems. Consider the following example: a human annotator, who is supervised by a human expert, performs a task of segmenting dogs in natural images. At some point the expert notices that the annotator makes a consistent mistake by always segmenting muzzles as a part of dogs. The expert can correct the annotator by preparing 5-10 correctly segmented images of dogs in muzzles and showing them to the annotator. On the other hand, it probably would be more efficient to just verbally warn the annotator that "a muzzle should not be considered as a part of a dog". We hypothesize, that similar ideas of high-level supervision can cheaply boost performance of current weakly-supervised segmentation models.

We make a step into this direction in Chapter 3, where we introduce a methodology for improving segmentation quality of segmentation of co-occurring objects, which are problematic for weakly-supervised models. Our protocol relies on the distributed image representation, learned by a deep neural network-based model, and requires the expert to answer only a few binary and model-specific questions. We hope that future research will further investigate this direction and develop novel methodologies for intelligent interaction between human experts and segmentation models, which are being learned.

Probabilistic image modeling. In the second part of this thesis we studied models for unsupervised learning. We built our work on top of probabilistic autoregressive models and propose a new extended family of this type of models. Within the proposed family we introduced two new models: *Grayscale PixelCNN* and *Pyramid PixelCNN*. We demonstrated that *Grayscale PixelCNN* is better than previously proposed models at capturing high-level image details, while, at the time, delivering competitive quantitative results. We expect that improved sample quality results indicates that learned generative model better captures semantic properties of natural images, and leave thorough verification of this hypothesis for future work. We also show that, unlike previously proposed autoregressive models, *Pyramid PixelCNN* scales to modeling and sampling high-resolution images and produces globally-coherent image samples

of high perceptual quality.

We have also studied an application of unsupervised modeling and applied autoregressive image modeling techniques for solving the automatic image colorization task. As a result, we derived the state-of-the-art colorization model, which is capable of producing multiple plausible colorizations for a single grayscale image. By inspecting visual results we observe that a learned colorization model captures many semantic concepts, as, otherwise, it would be impossible to guess correct colors.

We believe that tremendous recent progress on natural image modeling creates many new meaningful applications of image modeling techniques and that these applications should guide further development of unsupervised image models.

One group of such applications includes a wide range of image manipulation tasks, such as denoising, restoration, debluring, inpainting, super-resolution, compression and many others. For instance, we study one particular application, namely automatic image colorization, in Chapter 5. Unlike pure image modeling task, where it is inherently hard to define meaningful quantitative objective, these applications can be, in principle, judged by their performance as measured by domain-specific metrics. These metrics can be used as a principled quantitative guidance for future research in unsupervised image modeling.

Another interesting group of image modeling applications is related to unsupervised representation learning. There is a large amount of evidence, that image modeling techniques, explicitly or implicitly, learn visual representations that capture complex semantic properties of natural images. For instance, [40] demonstrates that unsupervised pretraining with raw images is helpful for solving object detection task.

However, it is unlikely that pure unsupervised image modeling techniques will be able to learn truly strong semantic representations from raw visual data alone. The reason is that the notion of "semantic" is subjective and depends on artificially defined semantic concepts. Thus, interesting future direction is to design unsupervised models augmented by additional guidance, which encourages them to better capture semantic concepts in natural images. For instance, we make effort into this direction by deriving *Grayscale PixelCNN* model, which, in comparison to a baseline model, has an additional built-in incentive to model semantic information in images.

Perhaps, it is beneficial to investigate data-oriented approaches for improving representations, which are learned by unsupervised models. The promising direction is to try to exploit external sources of additional information, which may be correlated with semantic content in images. For instance, if an image database is collected from the World-Wide-Web, then many images should have associated text appearing on the same web-page. Or, if visual data is collected by robots, then all images have associated measurements, which were taken by robot sensors at the moment when visual data was acquired.

Bibliography

[1] B. Alexe, T. Deselaers, and V. Ferrari. What is an object? In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 73–80, 2010.

[2] S. Andrews, I. Tsochantaridis, and T. Hofmann. Support vector machines for multiple-instance learning. In *Conference on Neural Information Processing Systems (NIPS)*, 2002.

[3] P. Arbeláez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[4] S. Arora and Y. Zhang. Do gans actually learn the distribution? an empirical study. *International Conference on Learning Representations (ICLR)*, 2018.

[5] P. Bachman. An architecture for deep, hierarchical generative models. In *Conference on Neural Information Processing Systems (NIPS)*, 2016.

[6] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*. Springer, 2006.

[7] L. Bazzani, A. Bergamo, D. Anguelov, and L. Torresani. Self-taught object localization with deep networks. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016.

[8] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei. What's the point: Semantic segmentation with point supervision. *European Conference on Computer Vision (ECCV)*, 2016.

[9] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Conference on Neural Information Processing Systems (NIPS)*, 2016.

[10] H. Bilen, M. Pedersoli, and T. Tuytelaars. Weakly supervised object detection with posterior regularization. In *British Machine Vision Conference (BMVC)*, 2014.

[11] H. Bilen, M. Pedersoli, and T. Tuytelaars. Weakly supervised object detection with convex clustering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[12] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[13] S. Branson, C. Wah, F. Schroff, B. Babenko, P. Welinder, P. Perona, and S. Belongie. Visual recognition with humans in the loop. In *European Conference on Computer Vision (ECCV)*, 2010.

[14] H. Caesar, J. Uijlings, and V. Ferrari. Coco-stuff: Thing and stuff classes in context. *arXiv preprint arXiv:1612.03716*, 2016.

[15] Y. Cao, Z. Zhou, W. Zhang, and Y. Yu. Unsupervised diverse colorization via generative adversarial networks. *arXiv preprint arXiv:1702.06674*, 2017.

[16] G. Carlsson, T. Ishkhanov, V. De Silva, and A. Zomorodian. On the local behavior of spaces of natural images. *International Journal of Computer Vision (IJCV)*, 76(1):1–12, 2008.

[17] J. Carreira and C. Sminchisescu. CPMC: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 34(7), 2012.

[18] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[19] G. Charpiat, M. Hofmann, and B. Schölkopf. Automatic image colorization via multimodal predictions. In *European Conference on Computer Vision (ECCV)*, 2008.

[20] A. Chaudhry, P. K. Dokania, and P. H. Torr. Discovering class-specific pixels for weakly-supervised semantic segmentation. *British Machine Vision Conference (BMVC)*, 2017.

[21] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *International Conference on Learning Representations (ICLR)*, 2015.

[22] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 40, 2018.

[23] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel. Variational lossy autoencoder. *International Conference on Learning Representations (ICLR)*, 2017.

[24] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. H. S. Torr. BING: Binarized normed gradients for objectness estimation at 300fps. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[25] R. G. Cinbis, J. Verbeek, and C. Schmid. Multi-fold MIL training for weakly supervised object localization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[26] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In *International Conference on Learning Representations (ICLR)*, 2016.

[27] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 1989.

[28] R. Dahl, M. Norouzi, and J. Shlens. Pixel recursive super resolution. *arXiv preprint arXiv:1702.00783*, 2017.

[29] J. Dai, K. He, and J. Sun. BoxSup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *International Conference on Computer Vision (ICCV)*, 2015.

[30] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[31] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, 1977.

[32] J. Deng, J. Krause, and L. Fei-Fei. Fine-grained crowdsourcing for fine-grained recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[33] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2015.

[34] T. Deselaers, B. Alexe, and V. Ferrari. Localizing objects while learning their appearance. In *European Conference on Computer Vision (ECCV)*, 2010.

[35] A. Deshpande, J. Lu, M. Yeh, and D. A. Forsyth. Learning diverse image colorization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[36] A. Deshpande, J. Rock, and D. A. Forsyth. Learning large-scale automatic image colorization. In *International Conference on Computer Vision (ICCV)*, 2015.

[37] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence*, 1997.

[38] L. Dinh, D. Krueger, and Y. Bengio. NICE: Non-linear independent components estimation. *International Conference on Learning Representations Workshop (ICLR workshop)*, 2015.

[39] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. *International Conference on Learning Representations (ICLR)*, 2017.

[40] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[41] J. L. Elman, E. A. Bates, M. H. Johnson, A. Karmiloff-Smith, D. Parisi, and K. Plunkett. *Rethinking Innateness: A Connectionist Perspective on Development*. MIT Press, 1996.

[42] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision (IJCV)*, 88(2), 2010.

[43] A. Freytag, E. Rodner, and J. Denzler. Selecting influential examples: Active learning with expected model output changes. In *European Conference on Computer Vision (ECCV)*, 2014.

[44] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 6(6):721–741, 1984.

[45] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[46] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2014.

[47] L. Gorelick, O. Veksler, Y. Boykov, and C. Nieuwenhuis. Convexity shape prior for segmentation. In *European Conference on Computer Vision (ECCV)*. Springer, 2014.

[48] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[49] K. Gregor, F. Besse, D. Jimenez Rezende, I. Danihelka, and D. Wierstra. Towards conceptual compression. In *Conference on Neural Information Processing Systems (NIPS)*, 2016.

[50] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learing (ICML)*, 2015.

[51] S. Guadarrama, R. Dahl, D. Bieber, M. Norouzi, J. Shlens, and K. Murphy. PixColor: Pixel recursive colorization. In *British Machine Vision Conference (BMVC)*, 2017.

[52] I. Gulrajani, K. Kumar, F. Ahmed, A. A. Taiga, F. Visin, D. Vazquez, and A. Courville. PixelVAE: A latent variable model for natural images. *International Conference on Learning Representations (ICLR)*, 2017.

[53] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *International Conference on Computer Vision (ICCV)*, 2011.

[54] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[55] X. He and R. S. Zemel. Learning hybrid models for image annotation with partially labeled data. In *Conference on Neural Information Processing Systems (NIPS)*, 2009.

[56] S. Hong, J. Oh, H. Lee, and B. Han. Learning transferrable knowledge for semantic segmentation with deep convolutional neural network. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[57] A. Hyvärinen, J. Hurri, and P. O. Hoyer. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision.*, volume 39. Springer, 2009.

[58] S. Iizuka, E. Simo-Serra, and H. Ishikawa. Let there be color!: Joint end-to-end learning of global and local image priors for automatic image colorization with simultaneous classification. *ACM Transactions on Graphics*, 2016.

[59] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learing (ICML)*, 2015.

[60] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[61] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093v1*, 2014.

[62] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[63] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with gaussian processes for object categorization. In *International Conference on Computer Vision (ICCV)*, 2007.

[64] M. Kawulok and B. Smolka. Competitive image colorization. In *IEEE International Conference on Image Processing*, 2010.

[65] H. Kim and S. Hwang. Scale-invariant feature learning using deconvolutional neural networks for weakly-supervised semantic segmentation. *arXiv preprint arXiv:1602.04984v2*, 2016.

[66] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2014.

[67] D. P. Kingma, T. Salimans, and M. Welling. Improving variational inference with inverse autoregressive flow. *Conference on Neural Information Processing Systems (NIPS)*, 2016.

[68] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[69] A. Kolesnikov, M. Guillaumin, V. Ferrari, and C. H. Lampert. Closed-form approximate CRF training for scalable image segmentation. In *European Conference on Computer Vision (ECCV)*, 2014.

[70] A. Kolesnikov and C. H. Lampert. Improving weakly-supervised object localization by micro-annotation. *British Machine Vision Conference (BMVC)*, 2016.

[71] A. Kolesnikov and C. H. Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. *arXiv preprint arXiv:1603.06098*, 2016.

[72] A. Kolesnikov and C. H. Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. *European Conference on Computer Vision (ECCV)*, 2016.

[73] A. Kolesnikov and C. H. Lampert. PixelCNN models with auxiliary variables for natural image modeling. In *International Conference on Machine Learing (ICML)*, 2017.

[74] P. Krähenbühl and V. Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *Conference on Neural Information Processing Systems (NIPS)*, 2011.

[75] J. Krapac and S. Šegvic. Weakly-supervised semantic segmentation by redistributing region scores to pixels. *German Conference on Pattern Recognition (GCPR)*, 2016.

[76] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Technical report, University of Toronto*, 2009.

[77] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2012.

[78] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *International Conference on Machine Learing (ICML)*, 2001.

[79] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 1950.

[80] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *European Conference on Computer Vision (ECCV)*, 2016.

[81] Q. V. Le, M. A. Ranzato, R. Monga, M. Devin, K. Chen, G. S. Corrado, J. Dean, and A. Ng. Building high-level features using large scale unsupervised learning. In *International Conference on Machine Learing (ICML)*, 2012.

[82] A. Levin, D. Lischinski, and Y. Weiss. Colorization using optimization. *ACM Transactions on Graphics*, 2004.

[83] S. Liu, S. Yan, T. Zhang, C. Xu, J. Liu, and H. Lu. Weakly supervised graph propagation towards collective image parsing. *IEEE Transactions on Multimedia (T-MM)*, 14(2), 2012.

[84] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *International Conference on Computer Vision (ICCV)*, 2015.

[85] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[86] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60, 2004.

[87] E. Lübbe. *Colours in the Mind - Colour Systems in Reality: A formula for colour saturation*. Books on Demand, 2010.

[88] W. Markle and B. Hunt. Coloring a black and white signal using motion detection, 1988. US Patent 4,755,870.

[89] Y. Morimoto, Y. Taguchi, and T. Naemura. Automatic colorization of grayscale images using multiple images on the web. *ACM Transactions on Graphics*, 2009.

[90] Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o (1/kˆ 2). In *Doklady AN USSR*, 1983.

[91] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision (ECCV)*, 2016.

[92] J. F. Novak. Method and apparatus for converting monochrome pictures to multi-color pictures electronically, 1972. US Patent 3,706,841.

[93] S. Nowozin, P. V. Gehler, and C. H. Lampert. On parameter learning in CRF-based approaches to object class image segmentation. In *European Conference on Computer Vision (ECCV)*, 2010.

[94] S. Nowozin and C. H. Lampert. Global connectivity potentials for random field models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[95] S. Nowozin and C. H. Lampert. Structured learning and prediction in computer vision. *Foundations and Trends in Computer Graphics and Vision*, 6(3–4), 2011.

[96] B. A. Olshausen and D. J. Field. Natural image statistics and efficient coding. *Network: computation in neural systems*, 7(2):333–339, 1996.

[97] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Is object localization for free? – weakly-supervised learning with convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[98] G. Ostrovski, M. G. Bellemare, A. v. d. Oord, and R. Munos. Count-based exploration with neural density models. *International Conference on Machine Learing (ICML)*, 2017.

[99] D. P. Papadopoulos, J. R. Uijlings, F. Keller, and V. Ferrari. We don't need no bounding-boxes: Training object class detectors using only human verification. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[100] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille. Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation. In *International Conference on Computer Vision (ICCV)*, 2015.

[101] A. Papazoglou and V. Ferrari. Fast object segmentation in unconstrained video. In *International Conference on Computer Vision (ICCV)*, 2013.

[102] D. Pathak, P. Krähenbühl, and T. Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *International Conference on Computer Vision (ICCV)*, 2015.

[103] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[104] D. Pathak, E. Shelhamer, J. Long, and T. Darrell. Fully convolutional multi-class multiple instance learning. *International Conference on Learning Representations Workshop (ICLR workshop)*, 2015.

[105] P. O. Pinheiro and R. Collobert. From image-level to pixel-level labeling with convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[106] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 1992.

[107] N. Pourian, S. Karthikeyan, and B. Manjunath. Weakly supervised graph based semantic segmentation by learning communities of image-parts. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[108] G.-J. Qi, X.-S. Hua, Y. Rui, J. Tang, and H.-J. Zhang. Two-dimensional active learning for image classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[109] A. Rabinovich, A. Vedaldi, C. Galleguillos, E. Wiewiora, and S. Belongie. Objects in context. In *International Conference on Computer Vision (ICCV)*, 2007.

[110] P. Ramachandran, T. L. Paine, P. Khorrami, M. Babaeizadeh, S. Chang, Y. Zhang, M. Hasegawa-Johnson, R. Campbell, and T. Huang. Fast generation for convolutional autoregressive models. *International Conference on Learning Representations Workshop (ICLR workshop)*, 2017.

[111] S.-A. Rebuffi, A. Kolesnikov, and C. H. Lampert. iCaRL: Incremental classifier and representation learning. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[112] S. Reed, A. v. d. Oord, N. Kalchbrenner, S. G. Colmenarejo, Z. Wang, D. Belov, and N. de Freitas. Parallel multiscale autoregressive density estimation. *International Conference on Machine Learing (ICML)*, 2017.

[113] S. E. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee. Learning what and where to draw. In *Conference on Neural Information Processing Systems (NIPS)*, 2016.

[114] H. Ringbauer, A. Kolesnikov, D. L. Field, and N. H. Barton. Estimating barriers to gene flow from distorted isolation by distance patterns. *Genetics*, 2018.

[115] S. Roth and M. J. Black. Fields of experts: A framework for learning image priors. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[116] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*. ACM, 2004.

[117] C. Rother, T. Minka, A. Blake, and V. Kolmogorov. Cosegmentation of image pairs by histogram matching-incorporating a global constraint into MRFs. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[118] A. Roy and S. Todorovic. Combining bottom-up, top-down, and smoothness cues for weakly supervised image segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[119] A. Royer, A. Kolesnikov, and C. H. Lampert. Probabilistic image colorization. *British Machine Vision Conference (BMVC)*, 2017.

[120] D. L. Ruderman and W. Bialek. Statistics of natural images: Scaling in the woods. *Physical review letters*, 73(6):814, 1994.

[121] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 2015.

[122] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015.

[123] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *Conference on Neural Information Processing Systems (NIPS)*, 2016.

[124] T. Salimans, A. Karpathy, X. Chen, D. P. Knigma, and Y. Bulatov. PixelCNN++: A PixelCNN implementation with discretized logistic mixture likelihood and other modifications. *International Conference on Learning Representations (ICLR)*, 2017.

[125] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Conference on Neural Information Processing Systems (NIPS)*, 2016.

[126] H. J. Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory (T-IT)*, 11(3), 1965.

[127] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.

[128] W. Shang, K. Sohn, D. Almeida, and H. Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *International Conference on Machine Learing (ICML)*, 2016.

[129] M. Shi and V. Ferrari. Weakly supervised object localization using size estimates. In *European Conference on Computer Vision (ECCV)*. Springer, 2016.

[130] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European Conference on Computer Vision (ECCV)*, 2006.

[131] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision (IJCV)*, 81, 2009.

[132] M. Simon and E. Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *International Conference on Computer Vision (ICCV)*, 2015.

[133] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. In *International Conference on Learning Representations (ICLR)*, 2014.

[134] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.

[135] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learing (ICML)*, 2015.

[136] H. O. Song, R. Girshick, S. Jegelka, J. Mairal, Z. Harchaoui, and T. Darrell. On learning to localize objects with minimal supervision. In *International Conference on Machine Learing (ICML)*, 2014.

[137] H. O. Song, Y. J. Lee, S. Jegelka, and T. Darrell. Weakly-supervised discovery of visual pattern configurations. In *Conference on Neural Information Processing Systems (NIPS)*, 2014.

[138] P. Tokmakov, K. Alahari, and C. Schmid. Weakly-supervised semantic segmentation using motion cues. In *European Conference on Computer Vision (ECCV)*. Springer, 2016.

[139] T. Toyoda and O. Hasegawa. Random field model for integration of local information and global information. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 30(8), 2008.

[140] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6, 2005.

[141] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, and A. Graves. Conditional image generation with pixelCNN decoders. In *Conference on Neural Information Processing Systems (NIPS)*, 2016.

[142] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *International Conference on Machine Learing (ICML)*, 2016.

[143] A. van den Oord and B. Schrauwen. Factoring variations in natural images with deep gaussian mixture models. In *Conference on Neural Information Processing Systems (NIPS)*, 2014.

[144] G. J. Van der Horst and M. A. Bouman. Spatiotemporal chromaticity discrimination. *Journal of the Optical Society of America (JOSA)*, 59(11):1482–1488, 1969.

[145] M. Vasconcelos, N. Vasconcelos, and G. Carneiro. Weakly supervised top-down image segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[146] J. Verbeek and B. Triggs. Region classification with Markov field aspect models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[147] J. Verbeek and W. Triggs. Scene segmentation with CRFs learned from partially labeled images. In *Conference on Neural Information Processing Systems (NIPS)*, 2008.

[148] A. Vezhnevets and J. M. Buhmann. Towards weakly supervised semantic segmentation by means of multiple instance and multitask learning. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[149] A. Vezhnevets, V. Ferrari, and J. M. Buhmann. Weakly supervised semantic segmentation with a multi-image model. In *International Conference on Computer Vision (ICCV)*, 2011.

[150] A. Vezhnevets, V. Ferrari, and J. M. Buhmann. Weakly supervised structured output learning for semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[151] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 2007.

[152] C. Wah, S. Branson, P. Perona, and S. Belongie. Multiclass recognition and part localization with humans in the loop. In *International Conference on Computer Vision (ICCV)*, 2011.

[153] C. Wah, G. Horn, S. Branson, S. Maji, P. Perona, and S. Belongie. Similarity comparisons for interactive fine-grained categorization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[154] C. Wang, K. Huang, W. Ren, J. Zhang, and S. Maybank. Large-scale weakly supervised object localization via latent category learning. *IEEE Transactions on Image Processing (T-IP)*, 2015.

[155] F. Wang, Q. Huang, M. Ovsjanikov, and L. J. Guibas. Unsupervised multi-class joint image segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[156] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. In *European Conference on Computer Vision (ECCV)*, 2016.

[157] Y. Wei, J. Feng, X. Liang, M.-M. Cheng, Y. Zhao, and S. Yan. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[158] Y. Wei, X. Liang, Y. Chen, Z. Jie, Y. Xiao, Y. Zhao, and S. Yan. Learning to segment with image-level annotations. *Pattern Recognition*, 2016.

[159] Y. Wei, X. Liang, Y. Chen, X. Shen, M. Cheng, Y. Zhao, and S. Yan. STC: a simple to complex framework for weakly-supervised semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*, 39(11), 2015.

[160] Z. Wu, C. Shen, and A. van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. arXiv:1611.10080, 2016.

[161] W. Xie, Y. Peng, and J. Xiao. Weakly-supervised image parsing via constructing semantic graphs and hypergraphs. In *ACM International Conference on Multimedia (MM)*, 2014.

[162] J. Xu, A. G. Schwing, and R. Urtasun. Tell me what you see and I will show you where it is. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[163] J. Xu, A. G. Schwing, and R. Urtasun. Learning to segment under various forms of weak supervision. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[164] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations (ICLR)*, 2016.

[165] L. Zhang, Y. Gao, Y. Xia, K. Lu, J. Shen, and R. Ji. Representative discovery of structure cues for weakly-supervised image segmentation. *IEEE Transactions on Multimedia (T-MM)*, 16(2), 2014.

[166] L. Zhang, M. Song, Z. Liu, X. Liu, J. Bu, and C. Chen. Probabilistic graphlet cut: Exploiting spatial structure cue for weakly supervised image segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[167] L. Zhang, Y. Yang, Y. Gao, Y. Yu, C. Wang, and X. Li. A probabilistic associative model for segmenting weakly supervised images. *IEEE Transactions on Image Processing (T-IP)*, 23(9), 2014.

[168] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *European Conference on Computer Vision (ECCV)*, 2016.

[169] W. Zhang, S. Zeng, D. Wang, and X. Xue. Weakly supervised semantic segmentation for social images. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[170] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *International Conference on Computer Vision (ICCV)*, 2015.

[171] B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[172] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene CNNs. In *International Conference on Learning Representations (ICLR)*, 2015.

[173] J. Zhu, J. Mao, and A. L. Yuille. Learning from weakly supervised data by the expectation loss SVM (e-SVM) algorithm. In *Conference on Neural Information Processing Systems (NIPS)*, 2014.

[174] S. C. Zhu, Y. Wu, and D. Mumford. Filters, random fields and maximum entropy (FRAME): Towards a unified theory for texture modeling. *International Journal of Computer Vision (IJCV)*, 27(2):107–126, 1998.

[175] D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *International Conference on Computer Vision (ICCV)*, 2011.