2019

Describing Images by Semantic Modeling using Attributes and Tags

Mahdi Mahmoudkalayeh
University of Central Florida

DESCRIBING IMAGES BY SEMANTIC MODELING USING ATTRIBUTES AND TAGS

by

MAHDI M. KALAYEH
M.S. Illinois Institute of Technology, 2010
B.S. Amirkabir University of Technology (Tehran Polytechnic), 2009

A dissertation submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy
in the Department of Computer Science
in the College of Engineering and Computer Science
at the University of Central Florida
Orlando, Florida

Spring Term
2019

Major Professor: Mubarak Shah

ABSTRACT

This dissertation addresses the problem of describing images using visual attributes and textual tags, a fundamental task that narrows down the semantic gap between the visual reasoning of humans and machines. Automatic image annotation assigns relevant textual tags to the images. In this dissertation, we propose a query-specific formulation based on Weighted Multi-view Non-negative Matrix Factorization to perform automatic image annotation. Our proposed technique seamlessly adapt to the changes in training data, naturally solves the problem of feature fusion and handles the challenge of the rare tags. Unlike tags, attributes are category-agnostic, hence their combination models an exponential number of semantic labels. Motivated by the fact that most attributes describe local properties, we propose exploiting localization cues, through semantic parsing of human face and body to improve person-related attribute prediction. We also demonstrate that image-level attribute labels can be effectively used as weak supervision for the task of semantic segmentation. Next, we analyze the Selfie images by utilizing tags and attributes. We collect the first large-scale Selfie dataset and annotate it with different attributes covering characteristics such as gender, age, race, facial gestures, and hairstyle. We then study the popularity and sentiments of the selfies given an estimated appearance of various semantic concepts. In brief, we automatically infer what makes a *good selfie*. Despite its extensive usage, the deep learning literature falls short in understanding the characteristics and behavior of the Batch Normalization. We conclude this dissertation by providing a fresh view, in light of information geometry and Fisher kernels to why the batch normalization works. We propose Mixture Normalization that disentangles modes of variation in the underlying distribution of the layer outputs and confirm that it effectively accelerates training of different batch-normalized architectures including Inception-V3, Densely Connected Networks, and Deep Convolutional Generative Adversarial Networks while achieving better generalization error.

EXTENDED ABSTRACT

The real world image databases such as Flickr, Tumblr, Google Images, Instagram or Facebook are characterized by continuous addition of new images. The recent approaches for automatic image annotation, *i.e.* the problem of assigning tags to images, have two major drawbacks. First, either models are learned using the entire training data, or to handle the issue of dataset imbalance, tag-specific discriminative models are trained. Such models become obsolete and require relearning when new images and tags are added to database. Second, the task of feature-fusion is typically dealt using ad-hoc approaches. In this dissertation, we propose a weighted extension of Multi-view Non-negative Matrix Factorization (NMF) to address the aforementioned drawbacks. The key idea is to learn query-specific generative model on the features of nearest-neighbors and tags where a consensus constraint is imposed on the coefficient matrices across different features. This results in coefficient vectors across features to be consistent and, thus, naturally solves the problem of feature fusion, while the weight matrices introduced in the proposed formulation alleviate the issue of dataset imbalance. Furthermore, our approach, being query-specific, is unaffected by addition of images and tags to the database.

Semantically describing visual content using tags has an inherent problem though. That is the number of concepts that a machine can describe is in one-to-one correspondence with its dictionary of learned textual tags. Instead, attributes provide a much more powerful tool. Attributes are semantically meaningful characteristics whose applicability widely crosses category boundaries (*e.g* “happy” can describe both a “dog” and a “person”), and their combinations allow describing an exponential number of concepts. They are particularly important in describing and recognizing concepts for which no explicit training example is given, *e.g., zero-shot learning*. Additionally, since attributes are human describable, they can be used for efficient human-computer interaction. Furthermore, attributes once decently modeled, have shown to assist more natural forms of description like image and video captioning. However, robustly detecting visual attributes is still far from being solved in computer vision. In this dissertation, we propose to employ semantic segmentation to improve person-related attribute prediction. The core idea lies in the fact that many attributes describe local properties. In other words, the probability of an attribute to appear in an image is far from being uniform in the spatial domain. For example, we expect one to focus on the “hair” region once detecting if someone “has wavy hair”; similarly to the “mouth” region if detecting “having goatee” is of interest. We build our attribute prediction model jointly with a deep semantic segmentation network. This harnesses the localization cues learned by the semantic segmentation to guide the attention of the attribute prediction to the regions where different attributes naturally show up. As a result of this approach, in addition to prediction, we are able to localize the attributes despite merely having access to image-level labels (weak supervision) during training. We first propose semantic segmentation-based pooling and gating, respectively denoted as SSP and SSG. In the former, the estimated segmentation masks are used to pool the final activations of the attribute prediction network, from multiple semantically homogeneous regions. This is in contrast to global average pooling which is agnostic with respect to where in the spatial domain activations occur. In SSG, the same idea is applied to the intermediate layers of the network. Specifically, we create multiple copies of the internal activations. In each copy, only values that fall within a certain semantic region are preserved while outside of that, activations are suppressed. This mechanism allows us to prevent pooling operation from blending activations that are associated with semantically different regions. SSP and SSG, while effective, impose heavy memory utilization since each channel of the activations is pooled/gated with *all* the semantic segmentation masks. To circumvent this, we propose Symbiotic Augmentation (SA), where we *learn* only one mask per activation channel. SA allows the model to either pick one, or combine (weighted superposition) multiple semantic maps, in order to generate the proper mask for each channel. SA simultaneously applies the same mechanism to the reverse problem by leveraging output logits of attribute prediction to guide the semantic segmentation task. We evaluate our proposed methods for facial attributes on CelebA and LFWA datasets, while benchmarking WIDER Attribute and Berkeley Attributes of People for whole body attributes. Our proposed methods achieve superior results compared to the previous works. Furthermore, we show that in the reverse problem, semantic face parsing significantly improves when its associated task is jointly learned, through our proposed Symbiotic Augmentation, with facial attribute prediction. We confirm that when few training instances are available, indeed image-level facial attribute labels can serve as an effective source of weak supervision to improve semantic face parsing. That reaffirms the need to jointly model these two interconnected tasks.

We can now exploit both tags, in form of concept scores, and attributes, to study how state-of-the-art computer vision techniques can facilitate understanding the Selfie phenomenon. This massive number of self-portrait images taken and shared on social media is revolutionizing the way people introduce themselves and the circle of their friends to the world. While taking photos of oneself can be seen simply as recording personal memories, the urge to share them with other people adds an exclusive sensation to the selfies. Due to the Big Data nature of selfies, it is nearly impossible to analyze them manually. In this dissertation, we provide, to the best of our knowledge, the first selfie dataset for research purposes with more than 46,000 images, annotated with 36 different attributes. We address interesting questions about selfies, including how appearance of certain objects, concepts and attributes influences the popularity of selfies. We also study the correlation between popularity and sentiment in selfie images. In a nutshell, from a large scale dataset, we automatically infer what makes a selfie a *good selfie*.

Finally, we change gear and shift our focus towards a better understanding of fundamentals of deep learning. This is extremely important as deep learning is the basis of almost all today's computer vision techniques, including those that are proposed in this dissertation. Particularly, we look into Batch Normalization (BN) [1], which is essential to effectively train state-of-the-art deep Convolutional Neural Networks (CNN). It normalizes the layer outputs during training using the statistics of each mini-batch. BN accelerates training procedure by allowing to safely utilize large learning rates and alleviates the need for careful initialization of the parameters. In this dissertation, we study BN from the viewpoint of Fisher kernels that arise from generative probability models. We show that assuming samples within a mini-batch are from the same probability density function, then BN is identical to the Fisher vector of a Gaussian distribution. That means batch normalizing transform can be explained in terms of kernels that naturally emerge from the probability density function that models the generative process of the underlying data distribution. Consequently, it promises higher discrimination power for the batch-normalized mini-batch. However, given the rectifying non-linearities employed in CNN architectures, distribution of the layer outputs shows an asymmetric characteristic. Therefore, in order for BN to fully benefit from the aforementioned properties, we propose approximating underlying data distribution not with one, but a mixture of Gaussian densities. Deriving Fisher vector for a Gaussian Mixture Model (GMM), reveals that batch normalization can be improved by independently normalizing with respect to the statistics of disentangled sub-populations. We refer to our proposed soft piecewise version of batch normalization as Mixture Normalization (MN). Through extensive set of experiments on CIFAR-10 and CIFAR-100, using both a 5-layers deep CNN and modern Inception-V3 architecture, we show that mixture normalization reduces required number of gradient updates to reach the maximum test accuracy of the batch-normalized model by $\sim$31%-47% across a variety of training scenarios. Replacing even a few BN modules with MN in the 48-layers deep Inception-V3 architecture is sufficient to not only obtain considerable training acceleration but also better final test accuracy. We show that similar observations are valid for 40 and 100-layers deep DenseNet architectures as well. We complement our study by evaluating the application of mixture normalization to the Generative Adversarial Networks (GANs), where “mode collapse” hinders the training process. We solely replace a few batch normalization layers in the generator with our mixture normalization. Our experiments using Deep Convolutional GAN (DCGAN) on CIFAR-10 show that mixture-normalized DCGAN not only provides an acceleration of $\sim$58% but also reaches lower (better) “Fréchet Inception Distance” (FID) of 33.35 compared to 37.56 of its batch-normalized counterpart.

To the one and only

who is friend of he who has no friend.

∼

To the love of my life

whose smile is the most beautiful thing in the world.

∼

To my beloved parents

whom I owe everything I am and will be.

∼

To my little brother

whom I miss so much.

∼

To my grandma

whom I just want to see one more time.

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Mubarak Shah for his great support throughout this journey. He taught me how to conduct research systematically, and reminded me to always maintain the highest expectations of myself. Also, I would like to thank Prof. Gita Sukthankar, Prof. Nazanin Rahnavard, and Prof. Teng Zhang for accepting to be a part of my committee, and their helpful guidance throughout the process of proposing and defending my dissertation. I am thankful to Dr. Rahul Sukthankar who, despite the short period of our collaboration, showed me that one's humbleness and knowledge must only simultaneously grow. I will always remember and cherish interesting conversations I had with Dr. Ulas Bagci over coffee. Finally, I would like to thank all of the past and present members of the Center for Research in Computer Vision (CRCV), Tonya LaPrarie, Kenneth Davis, Amir Roshan Zamir, Afshin Dehghan, Subhabrata Bhattacharya, Enrique Ortiz, Haroon Idrees, Nasim Souly, Gonzalo Vaca, Berkan Solmaz, Khurram Soomro, Waqas Sultani, Sarfaraz Hussein, Shayan Modiri, Shervin Ardeshir, Amir Mazaheri, Aidean Sharghi, Aisha Orooj Khan, Krishna Regmi, Dong Zhang, Yonatan Tariku, Ruben Villegas, and Emrah Basaran for their support, the good times, and the great memories.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

An extremely large number of images is being created at this very moment by different people from all around the world. And this only adds to what has been aggregated and preserved so far. Hence, it is very much expected to ask how are we going to represent, organize and search through our images? The scenario can be as simple as finding a particular photo on our cellphones from a backpacking trip, which we enthusiastically want to show it to a friend. Given the scale of the problem, we have no choice but to design algorithmic techniques which effectively harness the modern computation power. Meanwhile, to shrink the semantic gap and moving towards more seamless and natural human-computer interaction, we should prioritize models that provide a semantic description of images. Such perspective facilitates our understanding of how machines infer visual content and paves the way for explainability of well-performing but complex models. It is in this spirit that the current dissertation addresses the problem of semantically describing images. It mainly does so through the lens of *textual tags* and *visual attributes* as means for semantic modeling of visual content.

This dissertation contributes to semantic modeling of images by proposing: (1) an automatic image annotation framework that assigns relevant textual tags (*e.g.* “car”, “building”, “sunny”, “old”) to images, allowing for natural content-based search through a large and continuously growing collection of images, (2) effective person-related attribute prediction models that robustly detect the appearance of fine-grained visual traits (*e.g.* “big lips”, “mouth slightly open”, “wearing long sleeves”) in human images, (3) the first large-scale Selfie dataset, and exploring the limits of current algorithms to automatically analyze the popularity and sentiment of Selfies, which very well reflect the mood, preferences and interests of our society, (4) a simple yet effective approach to accelerate training of deep convolutional neural networks, as the backbone of all today’s computer vision frameworks.

1.1 Automatic Image Annotation

Image annotation refers to the task of assigning relevant tags to query images based on their visual content [13, 14]. The problem is difficult because an arbitrary image can capture a variety of visual concepts, each of which would require separate detection. Each image can be represented using multiple features which may be low-level, *e.g.* RGB histograms and HOG, or mid-level such as object concepts, *e.g.* human, dog, sky etc., or even high-level denoting the broader class to which the image belongs, *e.g.*, structures, animal, food. These different features capture different aspects or views[1] of the image, thereby, providing complementary information. However, since each feature represents the same image, they all capture the same underlying latent structure. That is, it is possible to transform feature vectors for each image so that the new representations, with respect to some pre-defined distance metric, are consistent across all the views.

Automatic image annotation is crucial for searchable databases like Flickr, Tumblr, Google Images, Instagram or Facebook. One of the key characteristics of real world databases is the continuous addition of new images, which contain new tags as well. Till 2011, 6 billion images had been uploaded to Flickr[2] while almost a quarter trillion images have been shared on Facebook[3] with a total of 300 million images uploaded every day. For a method to be practical for such databases, it has to rely on minimal training as the addition of new images and tags can render the learned models less effective over time. This holds true for both the methods that learn a direct mapping from features to tags [15, 16], or those that learn tag-specific discriminative models [14, 17, 18] where positive set is comprised of images which contain a particular tag and the negative set include images which do not have that tag.

[1] For consistency with Machine Learning literature, "views" means features in this Section.

[2] `http://tinyurl.com/q4zdshq`

[3] `http://tinyurl.com/pktnba2`

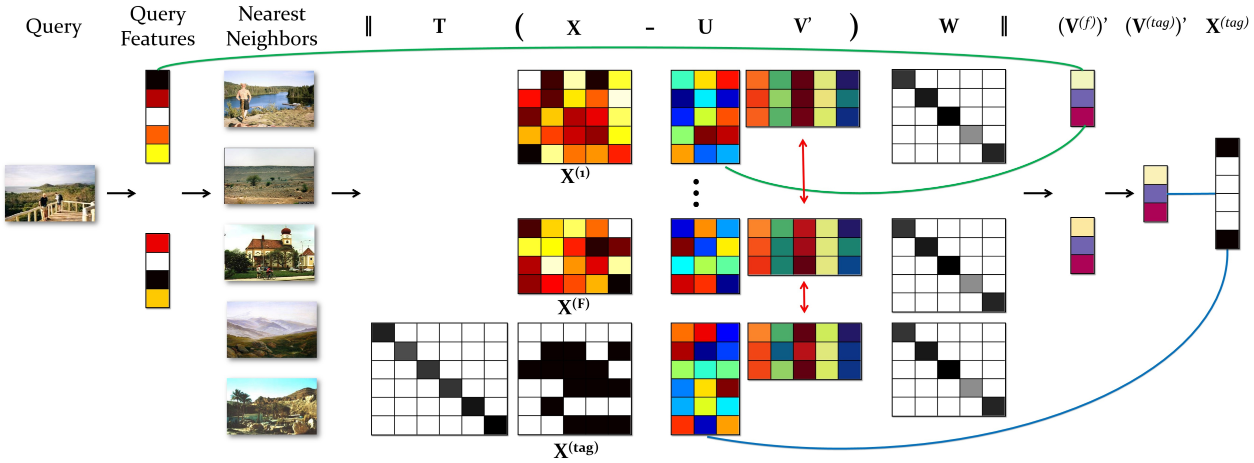


Figure 1.1: Schematic illustration of the proposed method: Given a query image, we extract different features which are used to find its nearest-neighbors. Then, using Non-negative Matrix Factorization using all the features $\mathbf{X}$, including tags, we find basis $\mathbf{U}$ and coefficient matrices $\mathbf{V}'$. The factorization for all matrices is done in a joint fashion by imposing a consensus constraint (red double arrows). Furthermore, to handle dataset imbalance, we introduce weight matrices $\mathbf{T}$ and $\mathbf{W}$ within the formulation. Using the basis matrices and corresponding features of the query, we find coefficient vector for each view (green lines). The matrix product of the tag-basis $\mathbf{U}^{(tag)}$ and mean of coefficient vectors from all views gives score for each tag (blue lines).

Obviously, as new images and tags are introduced into the database, the positive set for each tag will change, requiring retraining of the models. Inspired by the success of the recent nearest-neighbor approaches for image annotation [13, 14], we propose a novel method that learns a query-specific generative model using the nearest-neighbors. The proposed approach is illustrated in Figure 1.1.

The key idea of our approach is to treat tags as another view in addition to visual features, and find a joint factorization of all views into basis and coefficient matrices such that the coefficients of each training image are similar across views. This forces each basis vector to capture same latent concept in each view as well. After the factorization, the tags are transferred using both the model (basis) and the visual features of the query. Thus, given a query image, we first extract different visual features and find its nearest-neighbors. Then, using Non-negative Matrix Factorization on all views, *i.e.*, visual features as well as tags, we find basis and coefficient matrices. The coefficient vectors from all views, recovered using visual features of the query and the corresponding basis, are then averaged to get a unique coefficient vector. The matrix product of the final vector with basis for tags gives the scores for individual tags.

Non-negative Matrix Factorization [19] is a well-studied problem where the aim is to decompose a matrix into non-negative basis and coefficient matrices. The non-negative coefficients can then be seen as a soft assignment in terms of discovered basis [20, 21]. For the case of image annotation where multiple views are available, this problem requires NMF across all views. This is severely under-constrained and since the views capture the same latent structure, a consensus regularization can enforce the solution to discover a consistent latent structure for all views [22]. Each basis in each view then represents the same latent concept across views. In this work, we treat annotated tags as another view of the image and learn a set of basis across all views that correspond to the same underlying concepts. Note that, these concepts may not have any semantics associated with them, all that is implied is consistency in terms of the abstraction they capture.

One important issue in image annotation as encountered by all previous works is that of rare tags, those that do not occur frequently in the training data. For that, we introduce two weight matrices within the Multi-view NMF framework that increase the importance of both the rare tags and the images that contain rare tags. By assigning suitable weights, the NMF learns consistent latent concepts that are forced to capture the rare tags well, thus, allowing us to alleviate the issue of dataset imbalance by increasing recall for the rare tags. In summary, we propose to use Multi-view NMF for image annotation which learns a generative model specific to a particular query. The factorization is performed in such a way that ensures consistency in coefficients across features. This yields an elegant solution to the problem of feature fusion. Furthermore, we introduce weight matrices which increase the recall of the rare tags, without requiring tag-specific discriminative models. The proposed solution is practical for real world datasets characterized by continuous addition of images and tags.

Although effective, using tags to describe images imposes certain inherent limitations. Next, we explain these challenges, and will describe how attributes as sharable concepts can provide a better alternative. We focus on person-related attributes and propose to employ semantic face and body parsing to harness localization cues in order to improve attribute prediction task.

1.2 Person-related Attribute Prediction

Nowadays, state-of-the-art computer vision techniques allow us to teach machines different classes of objects, actions, scenes, and even fine-grained categories. However, to learn a certain notion, we usually need positive and negative examples from the concept of interest. This creates a set of challenges as the instances of different concepts are not equally easy to collect. Also, the number of learnable concepts is linearly capped by the cardinality of the training data. Therefore, being able to robustly learn a set of *sharable concepts* that go beyond rigid category boundaries is of tremendous importance. Visual attributes are one particular type of these *sharable concepts*. They are human describable and machine detectable. We can use attributes to describe a variety of objects, scenes, actions, and events. For example, we associate a person who is lying on a beach with the attribute *relaxed* or a cat that is chasing after a wool ball with the attribute *playing*.

Attributes are different from category labels in three major aspects. First, category labels are agnostic with respect to the intra-class variations that exist among different instances of a single category. Such flat representation cannot distinguish between a *grumpy* cat and a *joyful* one as it only sees them as cats.

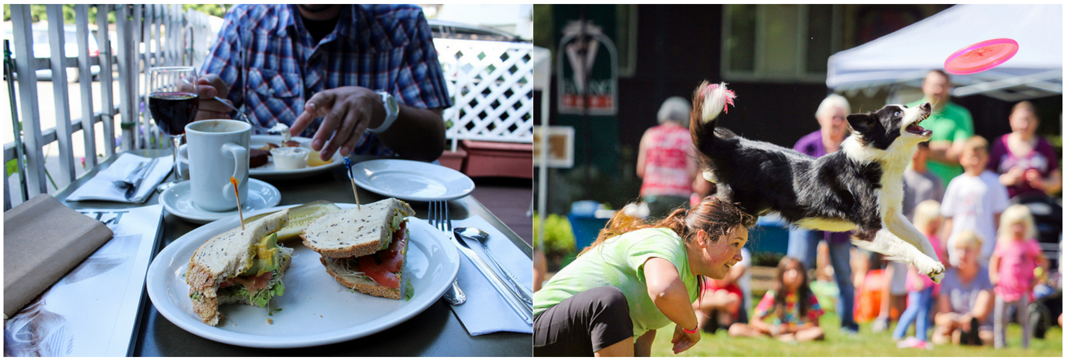


Figure 1.2: Examples of how contextual layout assists attribute prediction in wild. The *person* (on left) and the *dog* (on right) should be respectively labeled with the attributes *eating* and *catching*. This is hard to agree upon if we would have taken these object instances in isolation, out of their contexts *i.e* food and frisbee.

Second, attributes go across category boundaries. Hence, they can be used to potentially describe an exponential number of object categories (via different combinations) even if the associated category has never been observed before (*e.g* zero-shot learning). Third, unlike category labels that can be effectively inferred from the object itself, humans heavily rely on the contextual cues for the attribute prediction. Take the examples shown in Figure 1.2. If we only consider the bounding box around the dog, one would not assign *catching* to it. Instead, *running* may even be a valid attribute. However, leveraging contextual layout where the dog is floating in air, and close to a frisbee, provides human with sufficient indications to not only rule out the attribute *running* but also confidently label the dog with *catching*. Similarly, the table, food and plate, collectively serve as the context, building the ground to attribute the person with *eating*. Considering the aforementioned characteristics of attributes, we hypothesize that the attribute prediction task would benefit from contextual cues if they are properly represented. One can organize the context supervision into three levels: image-level, instance-level and pixel-level. Image-level supervision represents the context as a binary vector indicating whether an instance of a certain category appears somewhere in the context.

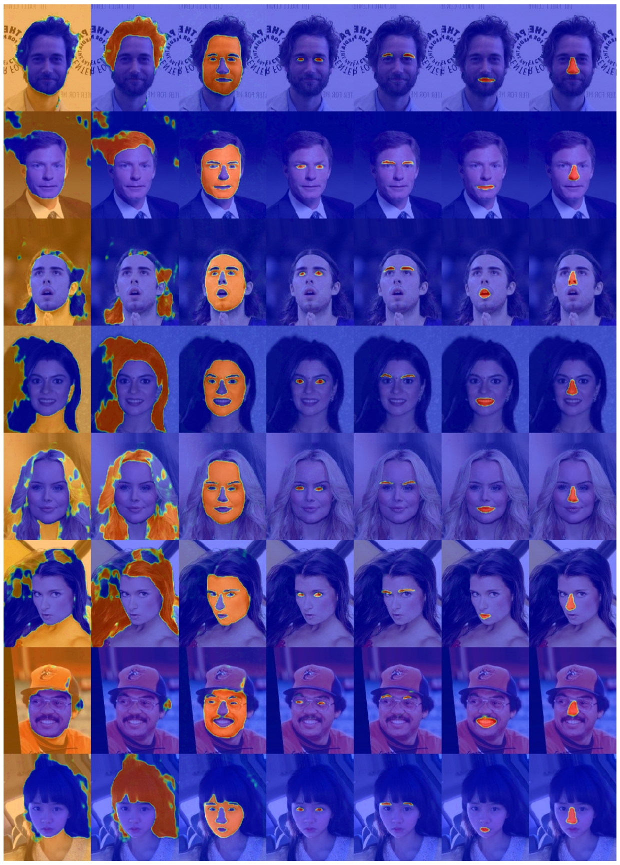


Figure 1.3: Examples of the segmentation masks generated by our semantic segmentation network [2] for previously unseen images. From left to right: background, hair, face skin, eyes, eyebrows, mouth and nose.

Therefore, it is blind to the spatial relationships that exist between underlying components *i.e* object instances in the scene. In the instance-level supervision, context is available in terms of a set of category label and bounding box tuples. That is, unlike the image-level, instance-level context supervision can model the spatial relationships in the scene. Lastly, in the pixel-level context supervision, we have access to the category labels in a per-pixel fashion. Obviously, this provides a much stronger supervision signal compared to the other two alternatives. In this work, we propose augmenting attribute prediction by transferring weakly pixel-level context supervision, from an auxiliary semantic segmentation task.

So far, we've explained attributes in general when they describe an instance of an object in a scene. However, the same is valid when attributes characterize variations of a certain object category. In this dissertation, we are interested in person-related, specifically facial and full body attributes. We view the concept of contextual cues, previously detailed for attributes of objects in the scene, as the natural correspondence of object attributes to the object parts and their associated layout in the spatial domain of the object boundary. Naturally, attributes are "additive" to the objects (*e.g.*, glasses for person). It means that an instance of an object may or may not take a certain attribute, while in either case the category label is preserved (*e.g.*, a person with or without glasses is still labeled as person). Hence, attributes are especially useful in problems that aim at modeling intra-category variations such as fine-grained classification. Despite their additive character, attributes do not appear in arbitrary regions of the objects (*e.g.*, hat if appears, is highly likely to show up on the top of person's head). This notion is the basis of our work. *We hypothesize that the attribute prediction can benefit from localization cues*. Specifically, to detect an attribute, instead of processing the entire spatial domain at once, one should focus on the region in which that attribute naturally shows up. However, not all attributes have precise correspondences. For example, it is ambiguous from where in the face, we as humans, infer if a person is *young* or *attractive*. Hence, instead of hard-coding the correspondences, even where those seem clear (*e.g.* glasses with nose and eyes), we allow the model to *learn* how to leverage the localization cues that are transferred from a relevant auxiliary task to the attribute prediction problem.

Using bounding boxes to show the boundary limits of objects is a common practice in computer vision. However, regions that different attributes are associated to drastically vary in terms of appearance. For example, in a face image, one cannot effectively put a bounding box around the region associated to "hair". In fact, the shape of the region can be used as an indicative signal on the attribute. On top of that, we have the partial occlusion of object parts which introduces further challenges by arbitrarily deforming visible regions. Therefore, we need an auxiliary task that learns detailed pixel-wise localization information without restricting the corresponding regions to be of certain pre-defined shapes. Semantic segmentation has all the aforementioned characteristics. It is the problem of assigning class labels to every pixel in an image. As a result, a successful semantic segmentation approach has to learn pixel-level localization cues which implicitly encode color, structure, and geometric characteristics in fine detail. In this dissertation, since we are interested in person-related attributes, we take face [4] and human body [23] semantic parsing problems as auxiliary tasks to steer the spatial focus of the attribute prediction methods accordingly.

To perform attribute prediction, we feed an image to a fully convolutional neural network which generates feature maps that are ready to be aggregated and passed to the classifier. However, global pooling [24] is agnostic to where, in spatial domain, the attribute-discriminative activations occur. Hence, instead of propagating the attribute signal to the entire spatial domain, we funnel it into the semantic regions. By doing so, our model learns *where* to attend and *how* to aggregate the feature map activations. We refer to this approach as Semantic Segmentation-based Pooling (SSP), where activations at the end of the attribute prediction pipeline are pooled within different semantic regions. Alternatively, we can incorporate the semantic regions into earlier layers of the attribute prediction network with a gating mechanism. Specifically, we propose augmenting max pooling operations such that they do not mix activations that reside in different semantic regions. Our approach generates multiple versions of the activation maps that are masked differently and presumably discriminative for various attributes. We refer to this approach as Semantic Segmentation-based Gating (SSG).

Since the semantic regions are not available for the attribute benchmarks, we learn to *estimate* them using a deep semantic segmentation network. For semantic face parsing [2], we take a conceptually similar approach to [25] in which an encoder-decoder model is built using convolution and deconvolution layers. However, considering the relatively small number of available data for the auxiliary segmentation task, we have to modify the network architecture. Despite being much simpler than [25], we found our semantic segmentation network [2] to be very effective in solving the auxiliary task of semantic face parsing. Examples of the segmentation masks generated for previously unseen images are illustrated in Figure 1.3. Once trained, such network is able to provide localization cues in the form of masks (decoder output) that decompose the spatial domain of an image into mutually exclusive semantic regions. We show that both SSP and SSG mechanisms outperform almost all the existing state-of-the-art facial attribute prediction techniques while employing them together results in further improvements.

One issue with SSP and SSG is their memory utilization. Since both architectures use the output of semantic segmentation to create K (referring to the number of semantic regions) copies of the previous convolution layer activations. Given limited GPU memory budget, this can restrict the application of these layers when K grows to large values. Instead, we can circumvent this by learning the proper mask per channel. In contrast to SSP and SSG which mask each and every channel of activations with *all* the K semantic probability maps, we propose to learn one mask per channel, as weighted superposition of different semantic probability maps (output of semantic segmentation network). Such workaround that can be simply implemented by a 1×1 convolution, adds minimum memory utilization overhead and also allows simplifying the SSP and SSG, yielding a single unified architecture which based on where it is applied in the architecture, mimics the behavior of SSP and SSG.

Following the recent trend in semantic segmentation, instead of an encoder-decoder approach, we then move to a fully convolutional architecture, specifically Inception-V3 [26]. Hence, we can unify attribute prediction and semantic segmentation networks by full weight sharing. This results in significant reduction in the number of model parameters and we no more need to pretrain the semantic segmentation network prior to deploying it in the attribute prediction pipeline. Instead, both tasks are learned simultaneously in an end-to-end fashion within a single architecture. We go beyond facial attributes and demonstrate the effectiveness of employing semantic segmentation in person-related attributes on multiple benchmarks (ref. Chapter 4). Furthermore, we provide comprehensive quantitative evaluation for the case where attributes are jointly trained with semantic segmentation with the aim to boost the latter task.

So far, we have covered how visual attributes and textual tags can be used to describe visual content in images. We also briefly introduced our proposed methods for automatic image annotation and person-related attribute prediction. Next, we explain how both tags, in form of concepts and object scores, along with attributes can automate understanding and facilitate describing Selfies, a global socio-cultural phenomenon, that is beyond traditional computer vision problems.

1.3 Describing Selfies

According to the Oxford Dictionary, *Selfie is a photograph that one has taken of oneself, typically one taken with a smartphone or webcam and shared via social media*. In the past few years, taking selfies has become very popular. People from different socio-economic, gender, race and age groups take selfies in various occasions. Google recently reported[4] that there are 93 million selfies taken every day only on Android devices. This gigantic data can reveal interesting statistics about the preferences, moods and feelings of the members of our society, if it is properly analyzed. Due to the large scale and continuous growth of the data, it is appropriate to employ machine learning and computer vision techniques to study selfie images. We assume that when someone takes a selfie, he/she considers two major aspects among many possible others to make it a *good selfie*, popularity and sentiment. Popularity refers to the log_2 normalized *view* counts while sentiment indicates the feeling that viewers infer by looking at a selfie. Typically, a social media user desires to increase the popularity of his/her selfie and imply a positive sentiment.

[4] http://goo.gl/53ZOjG

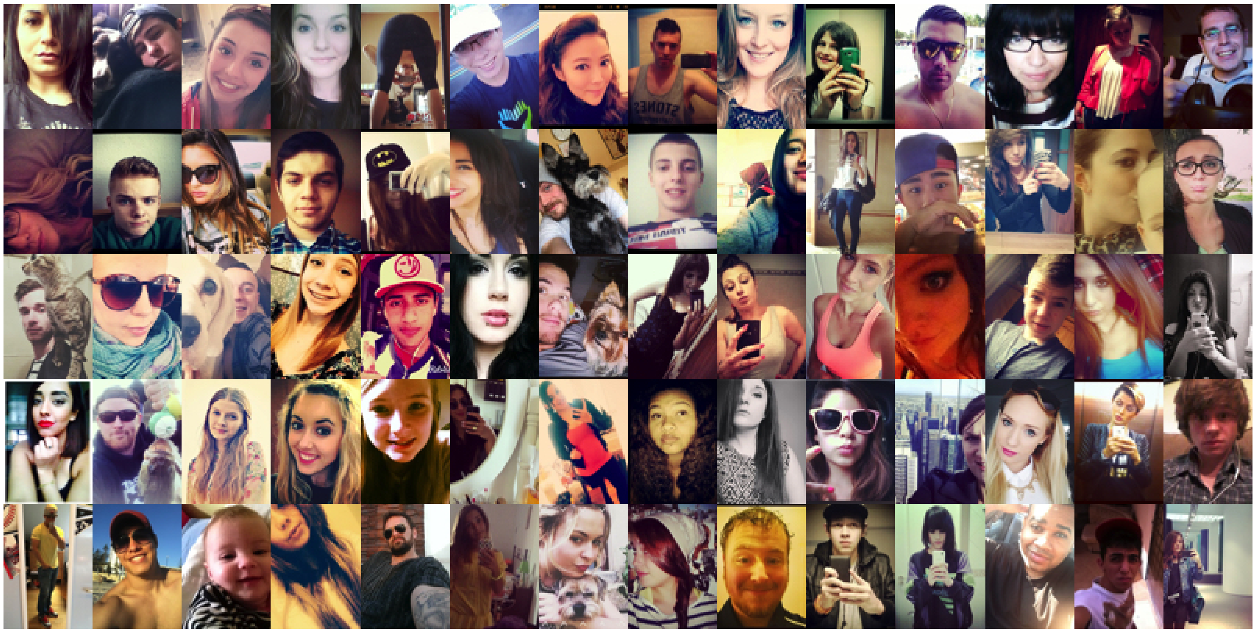


Figure 1.4: Examples of images in the Selfie dataset.

In this work, we introduce, to the best of our knowledge, the first selfie dataset for research purposes with more than 46,000 images, annotated with 36 different attributes (ref Figure 1.4). We also address the following questions: (1) How do different attributes, such as gender, race or hair color, influence the popularity of selfies? (2) How does the appearance of certain objects or particular concepts affect the popularity of selfies? (3) Is there a relationship between the sentiment inferred from a selfie and its popularity? (4) How does post-processing, such as applying different Instagram filters, influence the popularity of selfies?

Recently, Khosla et al. [27] proposed a framework to predict the popularity of a photograph before being uploaded on social media. They use about 2.3 million images from Flickr to predict the normalized number of views for images. Exploiting both visual content and social cues, Khosla et al. predict the popularity of an image with about 0.81 rank correlation to its ground truth. Borth et al. [28] have proposed sentiment prediction based on both visual content and tags associated to the images. They [28] use SentiBank, a library of visual concept detectors based on more than 2,000 Adjective Noun Pairs (ANP) to predict the implied emotion of a photograph. While these studies are valuable, they deal with photographs in general. We believe that studying selfies as the current most popular type of shared content on social media deserves an exclusive study. Knowing that photos with human faces are respectively 38% and 32% more likely to receive *likes* and *comments*, Bakhshi et al. [29], further demonstrate the importance of this work. Our study does not focus on improving the popularity score prediction or boosting the sentiment prediction performance. Instead, we are using state-of-the-art techniques in these areas to simply answer the four aforementioned questions.

Almost all today's deep convolutional neural architectures, including those that we propose in this dissertation, use Batch Normalization [1], yet the characteristics of it are not sufficiently studied in the literature. In the next section, we introduce a novel approach to the batch normalization [1] from the viewpoint of Fisher kernels [30] that suggests acceleration in training batch-normalized models can be achieved via disentangling modes of variation in the layer outputs.

1.4 Mixture Normalization

In the context of deep neural networks, the distribution of inputs to each layer is not only heavily dependent on the the previous layers but it also changes as the the network evolves through the training procedure. Ioffe and Szegedy [1] referred to this phenomenon as *internal covariate shift*. Another source of variation comes from the family of optimization techniques, specifically Stochastic Gradient Descent (SGD), that is used to train the deep neural networks. Although we can normalize a mini-batch at the input to the network, it will not remain normalized after multiple rounds of non-linear operations as we proceed from early layers to deeper ones. Batch Normalization (BN) [1] was proposed to alleviate the *internal covariate shift* by normalizing layer outputs for each training mini-batch with respect to its very own statistics, specifically mean and variance. Batch normalization has been successful in dramatically accelerating training procedure of deep neural networks and is now a standard component in almost all deep convolutional architectures. In fact, it is hard to imagine the possibility of effectively training state-of-the-art architectures such as Inception [26][31], Residual [32] and Densely connected [7] networks, with hundreds of layers, without employing batch normalization.

Batch normalization [1] and its few extensions [33][34][35][36], follow a general form (ref. Equation 2.6) to normalize a mini-batch, yet differ in the construction of the population over which mini-batch statistics are computed. In this dissertation, we provide a fresh view on the aforementioned general form of normalization, demonstrating its strong relation to the natural kernels that arise from generative probability models. Specifically, we show that assuming samples within a mini-batch are from the same probability density function, the general form of normalization is identical to the Fisher vector of a Gaussian distribution. Jaakkola and Haussler [30] proved that given classification labels as latent variables, Fisher kernel is asymptotically never inferior to the maximum *a posteriori* (MAP) decision rule. Therefore, the general form of normalization used in BN and its extensions, not only naturally emerges from the probability density function that models the generative process of the underlying data distribution, but also improves the discrimination power of the representation[5].

However, for BN [1] to truly enjoy theses properties, its input activations should follow a Gaussian distribution. We argue that due to the rectifying non-linearities employed in deep neural networks, it is unlikely that such condition is fully satisfied. Specifically, it is hard to believe that, in expectation, a linear combination (convolution operation) of multiple (different channels) distributions

[5] This is established on the basis of the discriminative derivations of Fisher kernel (ref. Theorem 1 of [30]). The mathematical derivations can be found in Appendix A of the longer version of [30], available at `https://people.csail.mit.edu/tommi/papers/gendisc.ps`.

with semi-infinite support (output of previous *e.g* ReLU [37]), with at least two modes of variation one for the rectified values mapped to zero and one for the positive values, results in a single Gaussian distribution. We visualize internal activations of both a shallow and very deep architecture, and observe that indeed corresponding activations very often illustrate asymmetric characteristic and are better approximated by a mixture model rather than a single Gaussian distribution. This observation builds the core of our work.

To equip batch normalization with characteristics which Fisher kernel promises, we must first properly approximate the probability density function of the internal representations. From [38], we know that any continuous distribution can be approximated with arbitrary precision using a Gaussian Mixture Model (GMM). Hence, instead of computing one set of statistical measures from the entire population (of instances in the mini-batch) as BN does, we propose normalization on sub-populations which can be identified by disentangling modes of the distribution, estimated via GMM. We refer to our proposed technique as Mixture Normalization (MN). While BN can only scale and/or shift the *whole* underlying probability density function, mixture normalization operates like a soft piecewise normalizing transform, capable of completely re-structuring the data distribution by independently scaling and/or shifting individual modes of distribution.

We show that mixture normalization can effectively accelerate its batch normalization [39] counterpart, through reducing required number of gradient updates to reach the maximum test accuracy of the batch normalized model by ∼**31%-47%**, across a variety of training scenarios on CIFAR-10 and CIFAR-100. Mixture normalization handles training in large learning rate regime considerably better than batch normalization, and in majority of cases reaches even a better final test accuracy. It is worth pointing out that such acceleration in training procedure is obtained by solely replacing *a few* batch normalization modules with mixture normalization along the depth of the architecture. This is true both for shallow and very deep and modern architectures such as Inception-V3 [26] and DenseNet [7].

1.5 Summary

In this Chapter, we introduced the automatic image annotation problem in Section 1.1 where the task is to assign relevant tags to the query images based on their visual content. We then explained the limitations of describing images using textual tags and discussed how attributes provide more powerful alternative in Section 1.2. This section was focused on person-related attribute prediction and provided a glimpse to our proposed method which exploits cues obtained via semantic face and body parsing. Section 1.3 presented application of tags and attributes in describing Selfie images. Finally, we introduced Mixture Normalization (MN) in Section 1.4. Our proposed view to the batch normalization [1] puts this widely-used deep learning component in a completely novel light and suggests how it can be improved in a diverse set of applications.

The rest of this dissertation is structured as follows:

In Chapter 2, we review existing literature on automatic image annotation, attribute prediction, semantic segmentation as well as batch normalization. In Chapter 3, we present our proposed approach for automatic image annotation based on weighted non-negative matrix factorization. In Chapter 4, we begin by detailing our novel methods which employ semantic segmentation to improve person-related attribute prediction. We then extend to its reverse problem where visual attributes are utilized to boost semantic segmentation. Chapter 5 explains how textual tags and visual attributes can be leveraged to computationally address the Selfie phenomenon in large-scale. Finally, in Chapter 6, we propose mixture normalization where disentangling modes of variation accelerates training batch-normalized deep convolutional neural networks. We conclude this dissertation in Chapter 7.

CHAPTER 2: LITERATURE REVIEW

Here, we provide comprehensive literature review for three computer vision tasks, namely automatic image annotation in Section 2.1, attribute prediction in Section 2.2, and semantic segmentation in Section 2.3. These are the major research problems, studied in this dissertation, which focus on semantically describing visual content in images. We then discuss batch normalization, a very widely-used component in all the modern deep neural network architectures. We detail in Section 2.4, how batch normalization and its various extensions are related through a general normalization formulation and what factors distinguish them from each other. These would build the ground for the next few chapters where we detail our novel methods to address the aforementioned problems.

2.1 Image Annotation

Over the past decade, significant efforts have been devoted to the task of image annotation. Many approaches are generative in nature consisting of either the topic or mixture models. In mixture model-based approaches, each annotated image is modeled as a mixture of topics over visual and tag features, where the mixture proportions are shared between different features or views. Examples include latent Dirichlet allocation [40], probabilistic latent semantic analysis [41], hierarchical Dirichlet processes [42], machine translation methods [43], and canonical correlation analysis [44]. The approach by Xiang et al. [45] also performs query-specific training using Markov random fields but it has expensive testing as one MRF is generated per tag. Mixture models define a joint distribution over image features and annotations. Given a query image, these models compute the conditional probability over tags given the visual features by marginalizing the joint likelihood. Carneiro et al. [46] use a fixed number of mixture components over visual features per tag, while in [47], it is defined by using the training images as components over visual features and tags.

Yavlinsky et al. [48] annotate images using only global features and perform nonparametric density estimation over the features. Besides generative approaches, discriminative models have also been used including SVM [49, 17], ranking SVM by Grangier et al. [50] and the method by Hertz et al. [51] which uses boosting.

A number of recent papers have reported better results with simple data-driven approaches by finding visually similar training images for a given query followed by transfer of tags from those images. Joint Equal Contribution (JEC) by Makadia et al. [13] was one of the first papers to highlight the effectiveness of the nearest-neighbors (NN) for image annotation. The paper presented an ad-hoc but simple procedure to transfer annotations from NN to the query image. The authors found that equal contributions from different features (mean of distances) performs on par with computationally expensive L_1-regularized Logistic Regression (Lasso). In contrast, we propose to fuse features using Multi-view NMF and show that it improves results. Guillaumin et al. [14] introduced TagProp which also uses nearest-neighbors to transfer tags. They showed that using large number of features, metric learning and special handling of rare tags (tag-specific models) improve results of image annotation. The nearest-neighbors are employed both during training and testing. Verma and Jawahar [52] presented two-pass kNN to find neighbors in semantic neighborhoods besides metric learning which learns weights for combining different features. The nearest-neighbor search they require for their method scales superlinearly with the number of training images, as a single image can occur in multiple semantic neighborhoods.

Non-negative matrix factorization has been successfully applied to various domains including text (document clustering [53]) and vision (face recognition [54]) and, in general, is an active area of research in clustering. Unlike PCA, the non-negativity of coefficients can be readily translated as weighted-assignment to basis or clusters. To understand the relationship between PCA, VQ and single-view NMF, the reader is referred to [55, 54]. The work by [56] proposes to integrate multiple views but the optimization is not performed jointly among views. They also propose a model selection strategy for identifying the correct number of clusters (basis). The work by Liu et al. [22] proposes a multi-view extension of NMF along with a novel normalization which makes all the basis to have unit sum permitting interpretation in terms of pLSA [20, 21]. Our approach [57] is a weighted extension of [22], and differs in three aspects. First, [22] uses Multi-view NMF for unsupervised data clustering, i.e. they assume all data is available. For the task of image annotation which is supervised multi-label concept detection, testing data is not known. Thus, we need to recover coefficients for the query image during testing. Second, in our formulation, we introduce weight matrices to handle imbalanced data. The third and most important difference is that, while [22] only uses visual features, we use tags as another feature and force Multi-view NMF to learn a set of basis across visual features and tags that are consistent across views. The key insight is that if we learn basis across features enforcing consistency on coefficients, then it is possible to use learned tag basis and query's visual features to obtain tags.

The proposed use of Multi-view NMF, introduced in Section 1.1, is also related to Relaxed Collaborative Representation [58], but rather than using features of training images directly as dictionaries, we learn a query-specific set of basis in each view and use that to transfer tags from annotated nearest-neighbors to the query image. One can also see our proposed approach as a multi-view extension with multiple weight matrices of the weighted but single-view NMF [59].

2.2 Attribute Prediction

Early works in modeling attributes [60][61][62] came around with the intention to change the recognition paradigm from naming objects to describing them. Therefore, instead of directly learning the object categories, one begins with learning a set of attributes that are shared among different categories. Object recognition can then be built upon the attribute scores. Hence, novel categories are seamlessly integrated, via attributes, with previously observed ones. This can be used to ameliorate label misalignment between train and test data. Considering the importance of human category, research in person-related attribute prediction [63][64][10][65][66][12] has flourished over the years. To perform attribute prediction, some of the previous works have invested in modeling the correlation among attributes [67][68][69][70], while others have focused on leveraging the category information [71][72][73]. There are also efforts to exploit the context [11].

Another way to view the attribute prediction literature is to divide it into holistic versus part-based methods. The common theme among the holistic models is to take the entire spatial domain into account when extracting features from images. On the other hand, part-based methods begin with an attribute-related part detection and then use the located parts, in isolation from the rest of spatial domain, to extract features. It has been shown that part-based models generally outperform the holistic methods. However, they are prone to the localization error as it can affect the quality of the extracted features. Although, there are works that have taken a hybrid approach benefiting from both the holistic and part-based cues [74][10]. Our proposed methods fall in between the two ends of the spectrum. While we process the image in a holistic fashion, we employ localization cues in form of pixel-level semantic representations.

Among earlier works we refer to [63][66][12][75] as successful examples of part-based attribute prediction models. More recently, in an effort to combine part-based models with deep learning, Zhang *et al.* [75] proposed PANDA, a pose-normalized convolutional neural network (CNN) to infer human attributes from images. PANDA employs poselets [12] to localize body parts and then extracts CNN features from the located regions. These features are later used to train SVM classifiers for attribute prediction. Inspired by [75], while seeking to also leverage the holistic cues, Gkioxari *et al.* [74] proposed a unified framework that benefits from both holistic and part-based models through utilizing a deep version of poselets [12] as part detectors. Liu *et al.* [10] have taken a relatively different approach. They show that pre-training on massive number of object categories and then fine-tuning on image level attributes is sufficiently effective in localizing the entire face region. Such weakly supervised method provides them with a localized region where they perform facial attribute prediction. In another part-based approach, Singh *et al.* [76] use spatial transformer networks [77] to localize the most relevant region associated to a given attribute. They encode such localization cue in a Siamese architecture to perform localization and ranking for relative attributes. Rudd *et al.* [78] have addressed the widely recognized data imbalance issue in attribute prediction, by introducing mixed objective optimization network (MOON). The proposed loss function mixes multiple task objectives with domain adaptive re-weighting of propagated loss. [9] and [79] are more examples of recent works that have tried similarly to address the class imbalance in the multi-label problem of attribute prediction. Li *et al.* have recently proposed lAndmark Free Face AttrIbute pRediction (AFFAIR) [80], a hierarchy of spatial transformation networks that initially crop and align the face region from the entire —assumed to be in the wild —input image and then localize relevant parts associated with different attributes. Separate neural network architectures then extract feature representations from global and part-based regions where their fusion is used to predict different facial attributes.

Our proposed person-related attribute detection, introduced in Section 1.2, employs semantic segmentation to capture local characteristics in images. Specifically, for facial attributes, we utilize semantic masks, obtained from a separate pre-trained semantic segmentation network, to gate and pool the activations, respectively at middle and the end of the attribute prediction architecture. We then extend and improve the proposed framework beyond faces, and to the full human body. Meanwhile, unlike facial attribute detection model that uses two separate networks for the main and auxiliary tasks, here we employ a heavy weight sharing strategy, unifying the semantic segmentation and attribute prediction architectures. This yields a significant drop in the computation cost of the framework. Next, we discuss the semantic segmentation literature.

2.3 Semantic Segmentation

Semantic segmentation can be seen as a dense pixel-level multi-class classification problem, where the spatial (spatio-temporal) domain of images (videos) is partitioned using fine contours (volumes) into clusters of pixels (voxels) with homogeneous class labels. Prior to the wide-spread popularity of deep convolutional neural networks (CNN), semantic segmentation used to be solved via traditional classifiers such as Support Vector Machine (SVM) or Random Forest applied to the super-pixels [81][82]. Conditional Random Field (CRF) was often used in these methods as the post processing technique to smooth the segmentation results, based on the assumption that pixels which fall within a certain vicinity, with similar color intensity, tend to be associated with the same class labels.

Among earlier efforts in using deep convolutional neural networks for semantic segmentation, we can refer to Ciresan *et. al* [83] work on automatic segmentation of neuronal structures in electron microscopy images. Although, since the number of classes was limited to only membrane and non-membrane, their problem in fact reduces to foreground detection task. Later, upon tremendous success of deep convolutional neural networks in image classification, researchers began designing semantic segmentation models on the top of CNN models, which were previously trained for other tasks, mainly image classification [84][85][86][87][88]. These methods, by leveraging supervised pre-training on strongly correlated tasks (*e.g.* often labels in two tasks have some overlap), were able to facilitate training procedure for semantic segmentation. However, such an adoption introduces its very own challenges.

Unlike image classification where the activations just before the classifier are flattened via fully connected layer or global average pooling, semantic segmentation task requires the spatial domain to be maintained, specifically the output segmentation maps should be at least of the same size as the input image. Fully Convolutional Networks[84] popularized CNN architectures for semantic segmentation. Long *et. al* [84] proposed transforming fully connected layers into convolution layers along with up-sampling intermediate and final activations, whose spatial domain have reduced due to pooling layers through the network architecture. These techniques enable a classification model to output segmentation maps of arbitrary size when operating on input images of any size. Almost all the subsequent state-of-the-art semantic segmentation methods adopted this paradigm. The performance of semantic segmentation task will be compromised if the spatial information is not well preserved through the network architecture. In contrast, architectures designed for image classification very often use pooling layers to aggregate the context activations while discarding the precise spatial coordinates. To alleviate this conceptual discrepancy, two different classes of architectures have evolved.

First is the encoder-decoder based approach [25] in which the encoder gradually reduces the spatial domain through successive convolution and pooling layers, to generate the bottleneck representation. Then the decoder recovers the spatial domain by applying multiple layers of deconvolution or convolution followed by up-sampling, to the aforementioned bottleneck representation. There are usually shortcut connections from the encoder to the decoder, leveraging details at multiple scales, in order to help decoder recovering fine characteristics more accurately. U-Net[89] SegNet[85], and RefineNet[90] are the popular architectures from this class.

The second class of architectures developed around the idea of Dilated or Atrous convolutions [86]. Specifically, one can avoid using pooling layers in order to preserve detailed spatial information, but this will dramatically increase the computation cost as the following layers must operate on larger activation maps. However, using Atrous convolution [86] with dilation rate equal to the stride of the avoided pooling layer, results in the exact same number of operations as the regular convolution operating on pooled activations[1]. In other words, dilated or Atrous convolution layer

[1] It is worth pointing out that while the computation cost remains the same, employing dilated convolution demands more memory since the size of activation maps remains intact.

allows for an exponential increase in effective receptive field without reducing the spatial resolution. In a series of works [91][87], Chen *et. al.* demonstrated how Atrous convolution and its multi-scale variation, namely Atrous spatial pyramid pooling (ASPP) can be utilized within the framework of fully convolutional neural networks to improve the performance of the semantic segmentation task. While in earlier efforts [87], Dense CRF [88] has been used, more recent works [91] have shown comparable results without using such post-processing technique.

Semantic segmentation can be applied at a finer granularity where instead of the entire scene, an object is semantically parsed into its parts. Among popular examples, readers are encouraged to refer to [4, 3, 92, 93] for face, [94, 95, 96, 97, 98, 99] for general objects, and [23, 100, 101, 102, 103, 104, 105, 106, 107] for human body and clothing semantic parsing.

In this dissertation, since we are interested in attributes describing human, when alluding to semantic segmentation, we specifically mean face and human body semantic parsing. To obtain localization cues from full human body, our semantic segmentation model is a fully convolutional neural network based on Inception-V3 [26] architecture, where following [87][91] we have also incorporated Atrous spatial pyramid pooling (ASPP). In addition to utilizing semantic parsing for person-related attribute prediction, we will provide results on semantic face parsing as well. We show that, training an attribute prediction network with image-level supervision can effectively serve as an initialization for semantic parsing task, when the the number of training instances is limited.

2.4 Batch Normalization

Let's consider $x \in \mathbb{R}^{N \times C \times H \times W}$, a 4-D activation tensor in a convolutional neural network where N, C, H and W are respectively the batch, channel, height and width axes. BN [1] computes the mini-batch mean ($\mu_{\mathscr{B}}$) and standard deviation ($\sigma_{\mathscr{B}}$), formulated in Equation 2.1, over the set $\mathscr{B} = \{x_{1\ldots m} : m \in [1,N] \times [1,H] \times [1,W]\}$, where x is flattened across all but channel axis, and ε is used for numerical stability.

$$\mu_{\mathscr{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i \quad \sigma_{\mathscr{B}} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathscr{B}})^2 + \varepsilon} \tag{2.1}$$

If we assume that samples within the mini-batch are from the same distribution, the transform $x \to \hat{x}$ shown in Equation 2.2, generates a zero mean and unit variance distribution. Then, BN uses learnable scale (γ) and shift (β) parameters to transform the normalized distribution into one with β mean and γ standard deviation.

$$\hat{x}_i = \frac{1}{\sigma_{\mathscr{B}}}(x_i - \mu_{\mathscr{B}}) \quad y_i = \gamma\hat{x}_i + \beta \tag{2.2}$$

Following Ioffe and Szegedy's [1] terminology, we refer to the transform

$$\mathrm{BN}_{\gamma,\beta} : x_{1\ldots m} \to y_{1\ldots m} \tag{2.3}$$

as the *Batch Normalizing Transform*. In most of its applications, BN normalizes the output of a convolution layer just before the non-linear activation function (*e.g* ReLU [37]), which separates two consecutive convolution layers. As we discussed before, normalized activations, $\hat{x}_{1\ldots m}$, under the assumption that $x_{1\ldots m}$ are from the same distribution, are of zero mean and unit variance. Hence, applying ReLU function to $\hat{x}_{1\ldots m}$ can be approximately seen as rectifying half of the distribution. This may not necessarily be the optimum case from the perspective of minimizing the objective function during training. That is why the *learnable* scale (γ) and more crucially shift (β) parameter are important, as they allow the training procedure to shape the behavior of the non-linearity and consequently the entire model. It is easy to see (ref. Equation 2.4) that BN enables the the model to alternate between two extreme cases of ignoring the non-linearity and completely clipping the activations. While in the former, the effective depth of the network reduces as two back-to-back linear operations can be seen as one, the latter prevents the activations from propagating to the next layer.

$$
\begin{aligned}
\big\{\forall\gamma\exists\eta \lim_{\beta\to+\eta} \mathrm{ReLU}\big(\mathrm{BN}_{\gamma,\beta}(x)\big) = \mathrm{BN}_{\gamma,\beta}(x) | 0 < \gamma, \eta \ll \infty\big\},\\
\big\{\forall\gamma\exists\eta \lim_{\beta\to-\eta} \mathrm{ReLU}\big(\mathrm{BN}_{\gamma,\beta}(x)\big) = 0 | 0 < \gamma, \eta \ll \infty\big\}.
\end{aligned}
\tag{2.4}
$$

The behavior of BN at inference is slightly different from training phase. To cope with potential discrepancy between distributions and dependency of each normalized activation to other instances in the mini-batch, BN accumulates a running average of the statistics at the training phase and uses them for normalizing mini-batches at inference. Hence, unlike training, where each mini-batch is normalized with respect to its very own statistics, all the mini-batches at inference use the same running average statistics for normalization while the scale and shift parameters are freezed. This workaround is effective when the size of the mini-batch is large, its instances are i.i.d. samples from training distribution and pre-computed statistics do not change [39]. However, in the absence of these conditions, estimation of mean and variance becomes less accurate at mini-batch level, hence affecting the running average statistics, and consequently results in performance degradation. To address these drawbacks, Ioffe [39] proposed *Batch Renormalization* where a per-dimension affine transformation is applied to the normalized activations as

$$
\frac{x_i - \mu}{\sigma} = \frac{x_i - \mu_{\mathscr{B}}}{\sigma_{\mathscr{B}}} \cdot r + d, \text{where} \, r = \frac{\sigma_{\mathscr{B}}}{\sigma}, d = \frac{\mu_{\mathscr{B}} - \mu}{\sigma}. \tag{2.5}
$$

Note that the parameters of the transformation, r and d, are not trainable, instead they, in expectation, compensate for the difference between per mini-batch and over-population statistics. If the expected values of the per mini-batch statistics match the moving average ones, then the affine transformation reduces to an Identity resulting in batch renormalization [39] to behave identical to the batch normalization [1].

Large batch training of the neural networks is very well motivated as it reduces the training time by effectively leveraging the parallel and/or distributed computation. Yet the common practices and previous empirical observations [108] used to suggest that large batch training would not generalize as well as small batch training. Recently, Hoffer *et. al* [109] showed that there is no inherent generalization gap associated with large batch training once the number of iterations and learning rate are properly adapted. However, they noticed that the dependency of statistics used by batch normalization [1], to the entire mini-batch, can affect the generalization of large batch training. To ameliorate that, Ghost Batch Normalization [109] was proposed, where a large batch is first scattered over multiple small virtual ("ghost") batches. Then, each sample is normalized with respect to the mean and standard deviation of the ghost batch to which it belongs to. At inference, normalization is done via a weighted running average that aggregates the statistics of all ghost batches during training.

Similarly, concerned with the dependency of batch-normalized activations on the entire mini-batch, Salimans *et. al* [110] proposed Weight Normalization, which operates on the weight vectors instead of activations. Hence, it does not introduce any dependencies between the samples within a mini-batch. At core, it decouples the norm from the direction of the weight vectors. Specifically, considering $\mathbf{w}$ as a weight vector in a standard artificial neural network, weigh normalization performs optimization over g and $\mathbf{v}$ using $\mathbf{w} = \frac{g}{\|\mathbf{v}\|}\mathbf{v}$ reparameterization. Salimans *et. al* [110] present weight normalization as a cheaper and less noisy approximation to the batch normalization [1] because first, convolutional neural networks usually have much fewer weights than activations and second, norm of $\mathbf{v}$ is non-stochatic but mini-batch statistics can have high variance for small batch sizes. Unfortunately, the guarantees that [110] provides on the activations and gradients do not extend to models with arbitrary non-linearities or when the architecture contains layers without weight normalization [39].

Recently, a few extensions on batch normalization have been proposed, specifically, Layer Normalization (LN) [33], Instance Normalization (IN) [34], Group Normalization (GN) [35], and Divisive Normalization (DN) [36]. In this section, we loosely adopt notations from [36] and [35] to show what distinguishes all the aforementioned methods is solely the set on which sample statistics are computed. This unifying view was initially presented by Ren *et al.*[36] in an attempt to describe the relationship between BN, LN, and DN. Here we further extend it to IN and GN as well.

Let's consider $i = (i_N, i_C, i_L)$ as a vector indexing the tensor of activations $x \in \mathbb{R}^{N \times C \times L}$ associated to a convolution layer, where the spatial domain has been flattened ($L = H \times W$). Then the general normalization, $x \to \hat{x}$, is in form of

$$v_i = x_i - \mathbb{E}_{\mathscr{B}_i}[x], \quad \hat{x}_i = \frac{v_i}{\sqrt{\mathbb{E}_{\mathscr{B}_i}[v^2] + \varepsilon}}, \tag{2.6}$$

while similar to BN, γ and β parameters can be further applied to the normalized activations. Before describing the extensions to the batch normalization using Equation 2.6, it is worth pointing out that given

$$\mathscr{B}_i = \{j : j_N \in [1, N], j_C \in [i_C], j_L \in [1, L]\}, \tag{2.7}$$

the general normalization form reduces to the original batch normalization formulated in Equations 2.1 and 2.2.

Layer Normalization (LN) was proposed by Ba *et al.* [33], where they remove the inter-dependency of batch-normalized activations to the entire mini-batch. Despite its effectiveness in recurrent networks, LN underperforms BN when applied to the convolution layers. That is because LN enforces the same distribution on the entire spatial domain *and* along channel axis which is not natural in case of convolution layers as the visual information can dramatically vary over the spatial domain. LN can be formulated as Equation 2.6 when

$$\mathscr{B}_i = \{j : j_N \in [i_N], j_C \in [1,C], j_L \in [1,L]\}. \tag{2.8}$$

Instance Normalization (IN) was proposed by Ulyanov *et al.* [34] for the problem of image style transfer [111]. While enjoying no inter-dependency to other samples in the mini-batch, IN [34] uses more relaxed conditions than LN [33], by computing the statistics only over the spatial domain generating different mean and standard deviations for each sample *and* each channel. IN can be formulated as Equation 2.6 when

$$\mathscr{B}_i = \{j : j_N \in [i_N], j_C \in [i_C], j_L \in [1,L]\}. \tag{2.9}$$

Group Normalization (GN) [35] is somewhere in between LN and IN. GN divides the channels into multiple groups ($G = 32$ by default), then computes the statistics along L axis but only within a subgroup of the channels. Therefore, when the number of groups matches the channel size ($G = C$), GN is identical to IN. On the other hand, when there is only one group ($G = 1$), GN reduces to LN. GN has shown to be effective on image classification, object detection and segmentation, when batch size is very small (2 and 4) while providing comparably good results with BN on typical batch sizes. GN can be formulated as Equation 2.6 when

$$\begin{aligned}\mathscr{B}_i = \{j : j_N \in [i_N], j_C \in [1,C], j_L \in [1,L] \mid \\ \lfloor \frac{j_C}{C/G} \rfloor = \lfloor \frac{i_C}{C/G} \rfloor\},\end{aligned} \tag{2.10}$$

where $\lfloor \frac{j_C}{C/G} \rfloor$ ensures that normalizing x_i is only influenced by the activation vectors which fall within the same group as x_i.

Divisive Normalization (DN) [36] can be seen as a local version of LN, where in normalizing x_i, instead of all the activation vectors within the same layer, only those that are in a certain vicinity of x_i contribute. This very well addresses the aforementioned drawback of LN, when applied to the convolution layer. DN is formulated slightly different from the aforementioned normalization methods, specifically as

$$v_i = x_i - \mathbb{E}_{\mathscr{A}_i}[x] \quad \hat{x}_i = \frac{v_i}{\sqrt{\rho^2 + \mathbb{E}_{\mathscr{B}_i}[v^2]}}, \tag{2.11}$$

when

$$\mathscr{A}_i = \{j : d(x_i, x_j) \leq R_{\mathscr{A}}\} \quad \mathscr{B}_i = \{j : d(v_i, v_j) \leq R_{\mathscr{B}}\}, \tag{2.12}$$

where d denotes an arbitrary distance between two hidden units, ρ is the normalizer bias, and R denotes the neighbourhood radius. Ren *et al.* [36] have shown how varying ρ would allow DN followed by ReLU to alternate within a wide range of non-linear behaviors. DN shows promising results both on convolutional and recurrent networks and can be easily implemented as a convolutional operator where $R_{\mathscr{A}}$ and $R_{\mathscr{B}}$ are determined by the kernel size.

2.5 Summary

This Chapter began with a detailed literature review to the problem of automatic images annotation in Section 2.1. We then provided a comprehensive survey on the attribute prediction and semantic segmentation, various settings for each problem along with a wide range of techniques that have been employed by the research community to address these tasks, respectively in Sections 2.2 and 2.3. Finally, we concluded this Chapter by offering a detailed review of batch normalization and its various extensions in Section 2.4.

CHAPTER 3: AUTOMATIC IMAGE ANNOTATION

The results of this Chapter have been published in the following paper:

Mahdi M. Kalayeh, Haroon Idrees, Mubarak Shah, *"NMF-KNN: Image Annotation using Weighted Multi-view Non-negative Matrix Factorization,"* in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp. 184–191.[57]

Automatic image annotation assigns relevant textual tags to the images. This task is of extreme importance when it comes to organizing large gallery of images or conducting content-based image retrieval. We propose a mathematical framework based on Non-negative Matrix Factorization to perform automatic image annotation. Our proposed technique can seamlessly adapt to the continuous growth of datasets, meanwhile being built on the features of nearest-neighbors and tags, it functions in a query-specific fashion with no training per se. It naturally solves the problem of feature fusion and handles the challenge of rare tags by introducing weight matrices that penalize for incorrect modeling of less frequent tags and images that are associated with them.

In this Chapter, we begin by formulating the weighted non-negative matrix factorization in Section 3.1. This includes the optimization steps and details of integrating tags as an additional view in Section 3.1.1. We then propose a mechanism in Section 3.1.2 to handle the rare tags followed by the process of recovering tags associated with the query image. Experimental results and computational complexity analysis will be then provided in Section 3.2 to conclude this chapter.

3.1 Methodology

Given a query image, we first find its nearest-neighbors in the database which are assumed to be annotated with tags. Each image is represented in terms of visual features which we treat as different views of the image. We also treat tags as another view by obtaining binary vectors with length equal to the vocabulary size of tags. Then, the matrices from all views are decomposed to obtain basis in each view such that the coefficient vector of each NN image is consistent across all views. This gives a query-specific generative model from which the tags of query image are generated using its visual features.

3.1.1 Weighted Multi-view Non-negative Matrix Factorization

Given a query image represented with multiple visual features, we compute its distance to images in the database in each view using pre-defined distance metrics (see Sec. 3.2). Next, the distance is normalized to lie between 0 and 1 for all the images for each view. Then, the distances across views are combined by taking their average and the N images with the smallest average distance are selected as nearest-neighbors.

Let $\mathbf{X}^{(f)} \in \mathbb{R}^{M_f \times N}$ represent the matrix obtained by horizontal concatenation of features vectors of length M_f from N images in f-th view, with a total of F views. Since we treat tags as another view, we let $\mathbf{X}^{(F+1)} = \mathbf{X}^{(tag)}$. The goal of Multi-view NMF is to decompose each $\mathbf{X}^{(f)}$ into a basis matrix $\mathbf{U}^{(f)} \in \mathbb{R}^{M_f \times K}$ and a coefficient matrix $\mathbf{V}^{(f)} \in \mathbb{R}^{N \times K}$, where the parameter K defines the number of basis or latent concepts in each view. The factorization is subjected to soft-consensus regularization which enforces coefficient vectors corresponding to each image to be similar to a consensus vector in all views. This also results in the basis vectors to capture similar contents in their respective views. This is particularly desirable for image annotation as the coefficient vectors, recovered using the basis and the visual features of the query, are comparable across views.

Furthermore, to improve predictability of rare tags, we introduce two weight matrices in Multi-view NMF formulation which bias the factorization towards improved reconstruction for rare tags. Weight matrix $\mathbf{T}^{(f)} \in \mathbb{R}^{M_{(f)} \times M_{(f)}}$ is identity for views corresponding to visual features. However, it is a diagonal matrix for tag-view and is used to increase weight of rare tags so that reconstruction is biased towards a solution which results in improved performance on such tags. The matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$ gives more weight to images containing rare tags and is applied to all views.

The objective function for Weighted Multi-view NMF which constitutes reconstruction and regularization terms is given by:

$$\begin{aligned} L = & \sum_{f=1}^{F+1} \|\mathbf{T}\big(\mathbf{X}^{(f)} - \mathbf{U}^{(f)}\mathbf{V}'^{(f)}\big)\mathbf{W}\|_F^2 \\ & + \sum_{f=1}^{F+1} \lambda_f \|\mathbf{W}'\big(\mathbf{V}^{(f)}\mathbf{Q}^{(f)} - \mathbf{V}^*\big)\|_F^2 \\ & \text{s.t. } \forall 1 \leq f \leq F+1, \quad \mathbf{U}^{(f)}, \mathbf{V}^{(f)}, \mathbf{V}^* \geq 0, \end{aligned} \tag{3.1}$$

where $(.)'$ denotes transpose operator. Since the factorization obtained by NMF is not unique, i.e., $\mathbf{U}\mathbf{V}' = \mathbf{U}\mathbf{Q}^{-1}\mathbf{Q}\mathbf{V}'$, we use the $\mathbf{Q}$ to normalize $\mathbf{U}$ so that each basis vector sums to 1, i.e., $\|\mathbf{U}_{:,i}\| = 1$. The diagonal matrix $\mathbf{Q}^{(f)} \in \mathbb{R}^{K \times K}$ is defined as:

$$\mathbf{Q}^{(f)} = \mathrm{Diag}\bigg(\sum_{m=1}^{M} \mathbf{U}_{m,1}^{(f)}, \sum_{m=1}^{M} \mathbf{U}_{m,2}^{(f)} \cdots \sum_{m=1}^{M} \mathbf{U}_{m,K}^{(f)}\bigg). \tag{3.2}$$

The minimization is performed through an iterative procedure. First, we minimize L over $\mathbf{U}^{(f)}$ and $\mathbf{V}^{(f)}$ keeping $\mathbf{V}^*$ fixed. In the next step, we minimize L over $\mathbf{V}^*$ keeping $\mathbf{U}^{(f)}$ and $\mathbf{V}^{(f)}$ fixed.

The procedure is repeated for a fixed number of iterations. Both $\mathbf{U}^{(f)}$ and $\mathbf{V}^{(f)}$ are initialized with non-negative values. Since the function is non-convex, optimization converges to a local minima.

Minimize over $\mathbf{U}^{(f)}$ and $\mathbf{V}^{(f)}$, given $\mathbf{V}^*$: With $\mathbf{V}^*$ fixed, $\mathbf{U}^{(f)}$ does not depend on $\mathbf{U}^{(f')}$ and $\mathbf{V}^{(f')}$ for $f' \neq f$. In the following treatment, we drop notation of feature for clarity. The objective function for a particular feature for fixed $\mathbf{V}^*$ is given by,

$$\|\mathbf{T}\big(\mathbf{X}-\mathbf{U}\mathbf{V}'\big)\mathbf{W}\|_F^2 + \lambda_f \|\mathbf{W}'\big(\mathbf{V}\mathbf{Q}-\mathbf{V}^*\big)\|_F^2 \qquad \text{s.t. } \mathbf{U}, \mathbf{V} \geq 0. \tag{3.3}$$

Compute $\mathbf{U}^{(f)}$, given $\mathbf{V}^{(f)}$ and $\mathbf{V}^*$: We obtain the following multiplicative update rule (similar to [22]):

$$\mathbf{U}_{i,k} \leftarrow \mathbf{U}_{i,k} \nabla_{\mathbf{U}_{i,k}} \text{ and } \nabla_{\mathbf{U}_{i,k}} = \frac{(\mathbf{T}'\mathbf{T}\mathbf{X}\mathbf{W}\mathbf{W}'\mathbf{V})_{i,k} + \lambda_f \sum_{n=1}^{N} \mathbf{W}_{n,n}^2 \mathbf{V}_{n,k} \mathbf{V}_{n,k}^*}{(\mathbf{T}'\mathbf{T}\mathbf{U}\mathbf{V}'\mathbf{W}\mathbf{W}'\mathbf{V})_{i,k} + \lambda_f \sum_{m=1}^{M} \mathbf{U}_{m,k} \sum_{n=1}^{N} \mathbf{W}_{n,n}^2 \mathbf{V}_{n,k}^2} \tag{3.4}$$

Compute $\mathbf{V}^{(f)}$, given $\mathbf{U}^{(f)}$ and $\mathbf{V}^*$: We obtain the following multiplicative update rule (similar to [22]):

$$\mathbf{V}_{j,k} \leftarrow \mathbf{V}_{j,k} \nabla_{\mathbf{V}_{j,k}} \text{ and } \nabla_{\mathbf{V}_{j,k}} = \frac{(\mathbf{W}\mathbf{W}'\mathbf{X}'\mathbf{T}'\mathbf{T}\mathbf{U})_{j,k} + \lambda_f \mathbf{W}_{j,j}^2 \mathbf{V}_{j,k}^*}{(\mathbf{W}\mathbf{W}'\mathbf{V}\mathbf{U}'\mathbf{T}'\mathbf{T}\mathbf{U})_{i,k} + \lambda_f \mathbf{W}_{j,j}^2 \mathbf{V}_{j,k}} \tag{3.5}$$

Minimize over $\mathbf{V}^*$, given $\mathbf{U}^{(f)}$ and $\mathbf{V}^{(f)}$: Once $\mathbf{U}^{(f)}$ and $\mathbf{V}^{(f)}$ have been updated for each view in a particular iteration, we take derivative of (3.1) w.r.t $\mathbf{V}^*$, set it equal to 0 and obtain the following closed-form solution to $\mathbf{V}^*$:

$$\mathbf{V}^* = \frac{\sum_{f=1}^{F} \lambda_f \mathbf{W}\mathbf{W}'\mathbf{V}^{(f)}\mathbf{Q}^{(f)}}{\sum_{f=1}^{F} \lambda_f \mathbf{W}\mathbf{W}'} \tag{3.6}$$

3.1.2 Boosting Mechanism for Rare Tags

Since rare tags appear with low frequency, they are overshadowed by frequent tags during training which leads to low recall for rare tags. The matrices $\mathbf{T}^{(tag)}$ and $\mathbf{W}$ introduced in the NMF improve recall and address the issue of dataset imbalance. Weight matrix $\mathbf{T}^{(tag)}$ is a diagonal matrix with $\mathbf{T}_{i,i}^{(tag)}$ set to 1/frequency of i-th tag in a query's neighborhood. $\mathbf{T}^{(tag)}$ penalizes inaccurate matrix factorization in Eq.3.1 severely for rare tags to ensure that the learned $\mathbf{U}^{(tag)}$ accurately models the rare tags. Thus, all tags contribute equally to the loss function. Furthermore, the diagonal matrix $\mathbf{W}$ is embedded in Eq.3.1 to bias the learned basis matrices towards a more accurate factorization of images with rare tags. $\mathbf{W}_{j,j}$ equals the summation of 1/frequency of tags of j-th NN example. The images annotated with rare tags are more important for an accurate generative model around a query as they capture co-occurrence of rare tags with the more frequent ones.

3.1.3 Recovering Tags of Query

Given the factorized basis using features and tags of nearest-neighbors of a particular query image, we recover the coefficients for visual features of the query in terms of learned basis using GPSR[112], which gives stable results than least squares especially when basis matrix is ill-conditioned. Next, the coefficient vectors from all views are combined using weighted average with weight for view f equal to λ_f to estimate the coefficient vector $\tilde{\mathbf{V}}^{(tag)}$. In our experiments, those were set to 0.01 for visual features and 1 for tags, so that basis for visual features are aligned with those of tags. Finally, the product of $\tilde{\mathbf{V}}^{(tag)}$ with the $\mathbf{U}^{(tag)}$ which was learned during training, i.e., $\mathbf{U}^{(tag)}(\tilde{\mathbf{V}}^{(tag)})'$ gives the scores for predicted tags. The desired number of tags can be obtained by ranking the tags according to the obtained scores.

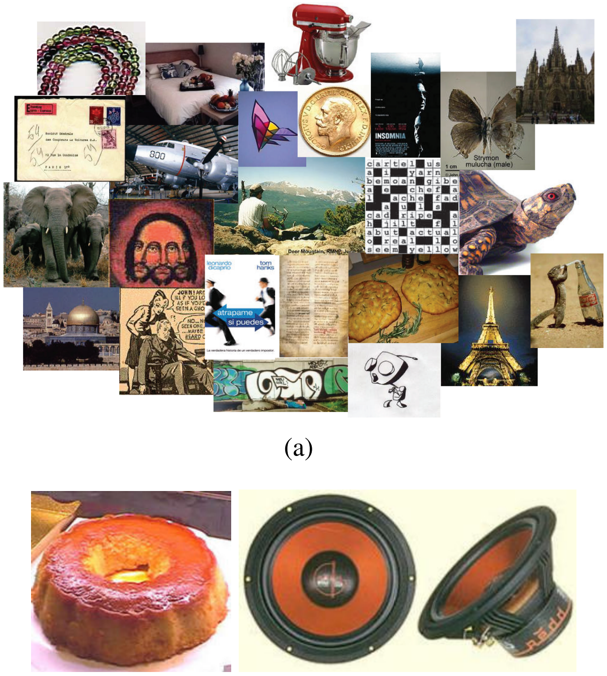
3.2 Experiments

In this section, we first explain the datasets used in our experiments as well as metrics used for evaluation. Next, we present the experimental results of proposed method on different datasets and compare them to previous works. Finally, we evaluate the effect of weight matrices and conclude the section with a brief discussion on the complexity of the proposed method.

3.2.1 Datasets, Evaluation Metrics and Features

We performed experiments on two popular and publicly available datasets Corel5K and ESP Game. Initially used by [43], Corel5K is the most common dataset for tag-based image annotation and retrieval. The training and testing sets consist of $4,500$ and 499 images, respectively. Images are manually annotated with, 3.4 tags on average, from a dictionary of 260 tags. ESP Game contains images annotated through an on-line game [113] in which players had to predict the same tags for images to gain points. Training and testing sets of ESP Game contain $18,689$ and $2,081$ images, respectively, with each image has 4.7 tags on average, from a dictionary of 268 tags. Figure 3.1 shows some example images from ESP Game dataset where we also illustrate the tag ambiguity for two images which share many tags while being visually and conceptually dissimilar.

We follow the evaluation metrics used in [14]. We automatically annotate each image with 5 tags and then compute precision and recall for each tag. The average precision (P) and average recall (R) across all tags in addition to the number of tags with non-zero recall ($N+$) is reported for performance evaluation. The F_1 measure, defined as harmonic mean of P and R ($F_1 = 2\frac{P*R}{P+R}$), is also reported.



(a)

(b) circle, orange, round

(c) circle, music, orange, red, round, white

Figure 3.1: Example images from ESP Game dataset are illustrated in 3.1(a). Figures 3.1(b) and 3.1(c) share many tags, although they are conceptually and visually different.

We used the publicly available features[1] provided by [14]. These are categorized as global and local descriptors. Global descriptors consist of GIST [114] and color histograms of RGB, Lab and HSV. Local descriptors include SIFT and robust (invariant to lighting geometry and specularities) hue descriptor [115] extracted around multi-scale grid and Harris-Laplacian interest points. Color histograms, SIFT and hue descriptors are also computed over three equal horizontal partitions (denoted by V3H1) for each image to encode spatial information. This provides a total of $F = 15$ features representing each image. To compare two features, we used L_1 for color histograms, L_2 for GIST and χ^2 for SIFT and hue descriptors, as was done in [14].

[1] Features: GIST, DenseSIFT, DenseSIFTV3H1, HarrisSIFT, HarrisSIFTV3H1, DenseHue, DenseHueV3H1, HarrisHue, HarrisHueV3H1, RGB, RGBV3H1, Lab, LabV3H1, HSV, HSVV3H1 - available at `http://tinyurl.com/l5d68sj`

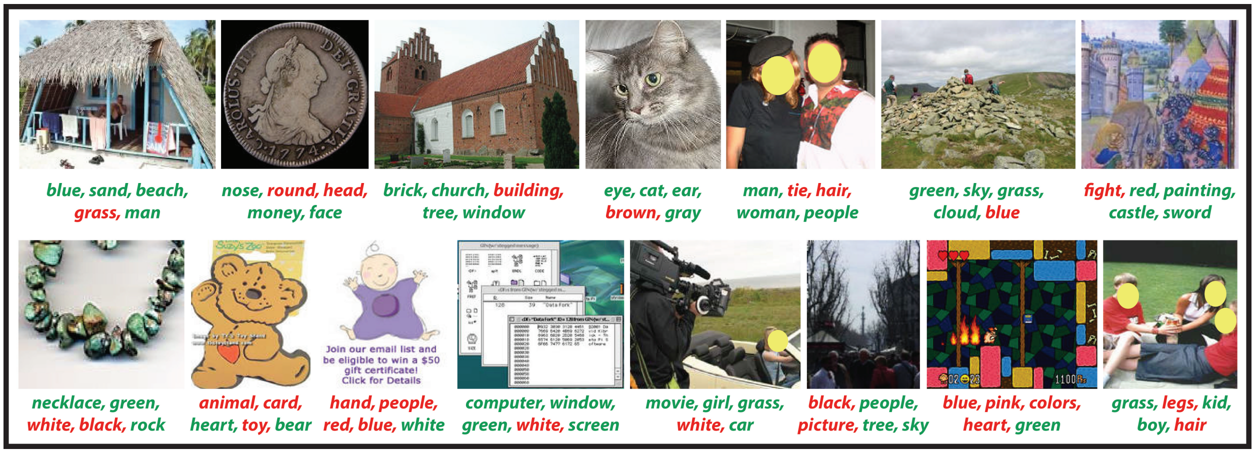

blue, sand, beach, grass, man
nose, round, head, money, face
brick, church, building, tree, window
eye, cat, ear, brown, gray
man, tie, hair, woman, people
green, sky, grass, cloud, blue
fight, red, painting, castle, sword
necklace, green, white, black, rock
animal, card, heart, toy, bear
Join our email list and be eligible to win a $50 gift certificate! Click for Details
hand, people, red, blue, white
computer, window, green, white, screen
movie, girl, grass, white, car
black, people, picture, tree, sky
blue, pink, colors, heart, green
grass, legs, kid, boy, hair

Figure 3.2: Example images from ESP Game dataset and the corresponding top 5 tags predicted using NMF-KNN are shown in this figure. Predicted tags in green appear in the ground truth while red ones do not. In many cases, even though the proposed method has predicted relevant tags to the image, those tags are missing in the ground truth. That is because the tag lists are not complete and are generally a subset of relevant tags.

3.2.2 Results

Table 3.1 compares the performance of the proposed NMF-KNN framework to existing approaches on Corel5k dataset. We can see that NMF-KNN significantly outperforms other image annotation algorithms including the ML variant of TagProp which does not use tag-specific discriminative models. This indicates that the proposed approach is more effective than weighted nearest-neighbor based approaches. To handle the issue of rare tags and boost their recall, TagProp [14] learns discriminant models for each tag given by the variant TagProp-σML, which is the state-of-the-art algorithm on this dataset. Although, TagProp-σML has a higher $N+$ with a difference of 10, the proposed method gives a much higher P, R, and F_1 making it competitive to TagProp-σML.

Table 3.2 shows the results of the proposed and comparison methods on ESP Game dataset. The proposed method provides competitive results w.r.t R and $N+$ to TagProp-σML. This shows that learning a model around a query is more useful and the natural capabilities of features fusion and handling rare tags, without requiring training on entire datasets, makes our approach superior to the previous methods. NMF-KNN performs slightly worse than FastTag [16], however, at constant time complexity during testing. Figure 3.2 illustrates qualitative results of image annotation using proposed method on some example images from ESP Game dataset. True positives, i.e., the tags predicted by our method that also occur in ground truth are shown in green, while false positives are shown in red. It is evident that many of the predicted tags are relevant to the image content, even though, they are not annotated in the ground truth. Another important difference between our method and existing methods [13, 14, 52] is the number of nearest-neighbors used to propagate the tags. These methods retrieve around $N = 200$ neighbors per query while we use only $N = 40$ neighbors. This suggests that NMF-KNN can build a reliable model around the query with 20% data compared to the competitive methods. To measure the effect of K, we evaluated the performance of NMF-KNN by increasing the value of K from 10 to 150 when N is fixed to 40. We observed that, for K beyond 50, R does not improve significantly while P initially increases and then reaches a plateau. Meanwhile, a larger K increases the computational complexity of the model and therefore is not desirable. We also studied the effect of neighborhood size, N, on the performance of our proposed method. For N larger than 40, we did not observe a considerable change in R, however P begins to decrease. A possible explanation is that for large neighborhood sizes, we allow irrelevant training examples to participate in construction of our query-specific model and therefore, learned basis become contaminated.

To evaluate the proposed boosting mechanism for rare tags, we study the effect of $\mathbf{W}$ and $\mathbf{T}$ in the Multi-view NMF. In Fig. 3.3(a) and 3.3(b), the y-axis shows the frequency with which tags appear in the training dataset. In Fig. 3.3(a), they have been grouped, while in Fig. 3.3(b), we show them individually. The x-axis shows the value of recall and the blue and yellow bars represent the before and after effect of $\mathbf{W}$ and $\mathbf{T}$, respectively.

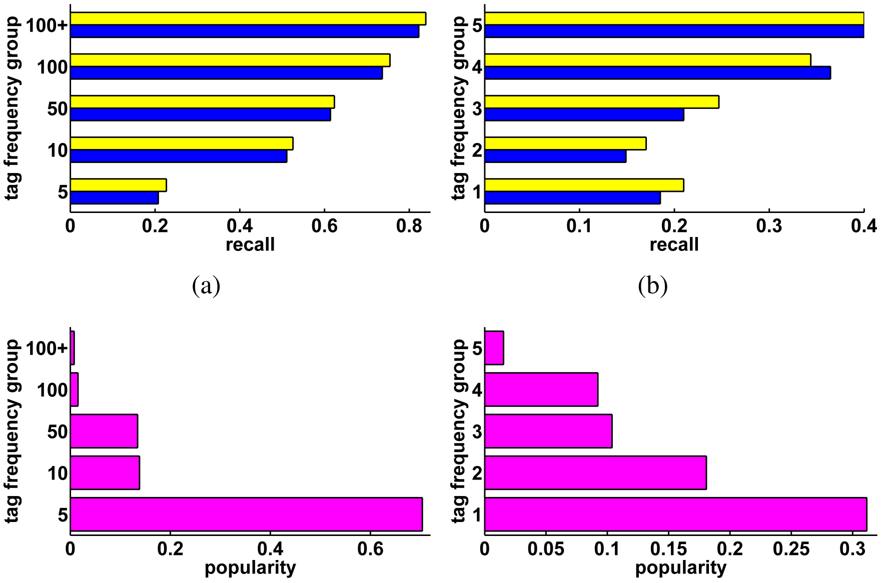
Table 3.1: Performance evaluation on Corel5k dataset

Method	P	R	F_1	$N+$
CRM[116]	16	19	17.3	107
InfNet[117]	17	24	19.9	112
NPDE[48]	18	21	19.3	114
SML[46]	23	29	25.6	137
MBRM[47]	24	25	24.4	122
TGLM[118]	25	29	26.8	131
JEC[13]	27	32	29.2	139
TagProp-ML[14]	31	37	33.7	146
TagProp-σML[14]	33	42	36.9	160
Group Sparsity[15]	30	33	31.4	146
FastTag[16]	32	43	36.7	166
NMF-KNN	38	56	45.2	150

Table 3.2: Performance evaluation on ESP Game dataset

Method	P	R	F_1	$N+$
MBRM[47]	18	19	18.4	209
JEC[13]	22	25	23.4	224
TagProp-σSD[14]	39	24	29.7	232
TagProp-ML[14]	49	20	28.4	213
TagProp-σML[14]	39	27	31.9	239
FastTag[16]	46	22	29.7	247
NMF-KNN	33	26	29.0	238

In Figure 3.3(c) and 3.3(d), the fraction of tags that belong to each group are shown. Boosting mechanism improves the mean recall of tags in five groups (Fig. 3.3(a)) by $\mathbf{1.91}\%$, $\mathbf{1.48}\%$, $\mathbf{0.94}\%$, $\mathbf{1.82}\%$ and $\mathbf{1.67}\%$, respectively. The first group contains tags with frequency less than 6 while the last one with frequency greater than 100. Fig. 3.3(c) shows that the majority ($\mathbf{70.38}\%$) of tags are assigned to less than 6 images in the dataset. From Fig. 3.3(d), we can see that tags with only 1 relevant image in the dataset are dominant. The proposed boosting mechanism increased the recall of tags with 1, 2 and 3 relevant tags in the dataset by $\mathbf{2.47}\%$, $\mathbf{2.13}\%$ and $\mathbf{3.70}\%$, respectively.



(a)

(b)

(c)

(d)

Figure 3.3: Evaluating the effect of Weight Matrices: Evaluated on Corel5k dataset, 3.3(a) shows the effect of using weight matrices, before (blue) and after (yellow), on the annotation performance. Tags are grouped based on their frequency of appearance in the dataset. The first bin groups words that have between 1 to 5 images related to them. The second bin is associated with tags with the images between 6 to 10, and so on. In 3.3(b) we show the same for first group of 3.3(a) to analyze the recall of tags with 1 to 5 related images. 3.3(c) and 3.3(d) give the fraction of tags in each bin of 3.3(a) and 3.3(b), respectively. This shows that we can improve the recall of rare tags without sacrificing that of frequent tags.

In summary, Fig. 3.3(a) shows that for the tags belonging to all frequencies, the boosting mechanism improves the recall. This is different from TagProp [14] which sacrifices recall of frequent tags to boost that of rare tags.

3.2.3 Computational Complexity

As noted by [119, 16, 15], [14]'s training complexity is quadratic, $O(n^2)$, where n is the number of training images. Since it relies on sophisticated training procedures and per tag optimizations, it is not scalable on large datasets. Adding new images or tags to the dataset influences the performance of trained models as both positive and negative instances change for discriminative classifiers. JEC [13] and FastTag [16] are comparable with proposed method in terms of complexity but [13] provides considerably lower performance. Since [16] performs a global co-regularized learning, regressor (W) and enricher (B) matrices have to be re-trained when a new set of samples or tags are introduced to the dataset.

The proposed Multi-view NMF framework does not require any training but has $O(n)$ test-time complexity due to nearest-neighbor look up for the query image where n is the total number of training examples. The complexity of Weighted Multi-view NMF is linear with respect to the cardinality of chosen nearest-neighborhood that results in $O(n)$ complexity for the proposed approach. In our experiments, query-specific training usually converges after $15-20$ iterations. The computation cost breakdown of NMF-KNN follows: 80% for finding the nearest-neighbors, 19% for learning the model and 1% for predicting tags. Sub-linear time complexity can be achieved by employing approximate NN search methods e.g. FLANN [120] or k-d trees in the implementation of NMF-KNN.

3.3 Summary

In this chapter, we addressed automatic image annotation task, the problem of assigning relevant textual keywords to images based on their visual content. Our proposed approach is suitable to real-world databases which are characterized by a continuous increase in both the images and tags assigned to those images. It is a hybrid of nearest-neighbor and generative approaches and does not require any training in the form of global mapping between features and tags or tag-specific discriminant models. NMF-KNN allows feature-fusion in a mathematically coherent way by discovering a set of basis using both the visual features and annotated tags. The proposed weight matrices handle the issue of dataset imbalance by increasing the recall of rare tags. Our proposed approach offers a practical solution to real world datasets and performs competitively with state-of-the-art methods without requiring any training. This is due to the proposed Weighted Multi-view NMF which learns a superior model specific to each query while requiring fewer number of nearest-neighbors for tag transfer.

Next, we detail our proposed technique for person-related attribute detection by harnessing localization cues obtained from semantic face and body parsing.

CHAPTER 4: ON SYMBIOSIS OF ATTRIBUTE PREDICTION AND SEMANTIC SEGMENTATION

The results of this Chapter have been partially published in the following paper:

Mahdi M. Kalayeh, Boqing Gong, Mubarak Shah, *"Improving Facial Attribute Prediction using Semantic Segmentation,"* in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 6942–6950.[2]

While attribute prediction measures the likelihood of multiple labels appearing in an image as a whole, semantic segmentation assigns class labels, in a multi-class classification fashion, to every single pixel in the image. We hypothesize that integrating pixel-level semantic parsing of the face and human body should improve person-related attribute prediction. In this regard, we propose Semantic Segmentation-based Pooling (SSP) and Gating (SSG). In SSP, the estimated segmentation masks pool the output activations of the last (before classifier) convolutional layer at multiple semantically homogeneous regions, unlike global average pooling that is spatially agnostic. In SSG, we create multiple copies where each preserves the activations within a single semantic region and suppresses otherwise. This mechanism prevents max-pooling from mixing semantically inconsistent regions. SSP and SSG while effective, impose heavy memory utilization. To tackle that, Symbiotic Augmentation (SA) is proposed, where we learn to generate only one mask per activation channel.

In this Chapter, we demonstrate the effectiveness of employing semantic segmentation to improve person-related attribute prediction. Specifically, we present semantic segmentation-based pooling (SSP) and gating (SSG), respectively in Section 4.1.1 and Section 4.1.2. Focusing on minimizing the memory utilization overhead, we then propose, in Section 4.1.3, a simple alternative architecture that approximately mimics the behavior of SSP and SSG, with considerably lower computation cost. We also unify semantic segmentation and attribute prediction in Section 4.1.6 through multi-tasking a single network and training it in an end-to-end fashion. Section 4.2 details our experiments where we achieve state-of-the-art results in person-related attribute prediction on CelebA, LFWA, WIDER Attributes, and Berkeley Attributes of People datasets. We conclude this Chapter in Section 4.2.5 by providing comprehensive experiments, detailing how to improve semantic segmentation task by leveraging image-level attribute annotations.

4.1 Methodology

The underlying idea of this work is to exploit semantic segmentation in order to improve person-related attribute prediction. To do so, we first propose semantic segmentation-based pooling (SSP) and gating (SSG). These two layers, when employed in a basic convolutional backbone, assist the attribute prediction by integrating detailed localization cues. The downside though is the additional computation cost. Hence, we follow by proposing a considerably simpler architecture, which unifies SSP and SSG designs while approximately mimicking their behavior with drastically smaller memory footprint. Furthermore, instead of two networks, one for semantic segmentation and the other for attribute prediction, we explore fully sharing the weights among two tasks, and train a single multi-tasking architecture in an end-to-end fashion. We then further improve our framework by replacing the convolutional backbone, initially similar to VGG-16 [121], with the more modern Inception-V3 [26]. This choice will allow us to further push performance boundaries in person-related attribute prediction task.

4.1.1 SSP: Semantic Segmentation-based Pooling

We argue that attributes usually have a natural correspondence to certain regions within the object boundary. Hence, aggregating the visual information from the entire spatial domain of an image would not capture this property. This is the case for the global average pooling used in our baseline as it is agnostic to where, in the spatial domain, activations occur. Instead of pooling from the entire activation map, we propose to first decompose the activations of the last convolution layer into different semantic regions and then aggregate only those that reside in the same region. Hence, rather than a single vector representation, we obtain multiple features, each representing only one semantic region. This approach has an interesting intuition behind it. In fact, SSP funnels the back-propagation of the label signals, via multiple paths, associated with different semantic regions, through the entire network. This is in contrast with global average pooling that rather equally affects different locations in the spatial domain. We later explore this by visualizing the activation maps of the final convolution layer.

We can simply concatenate the representations associated with different regions and pass it to the classifier; however, it is interesting to observe if attributes indeed prefer one semantic region to another. Also, whether what our model learns matches human expectation on what attribute corresponds to which region. To do so, we take a similar approach to [122] where Bilen and Vedaldi employed a two branch network for weakly supervised object detection. We pass the vector representations, each associated with a different semantic region, to two branches one for recognition and another for localization. We implement these branches as linear classifiers that map vector representations to the number of attributes. Hence, we have multiple detection scores for an attribute each inferred based on one and only one semantic region. To combine these detection scores, we normalize outputs of the localization branch using softmax non-linearity across different semantic regions. This is a per-attribute operation, not an across-attribute one. We then compute the final attribute detection scores by a weighted sum of the per-region logits (*i.e.* outputs of recognition branch) using weights generated by the localization branch. Figure 4.1 (Left) shows the SSP architecture.

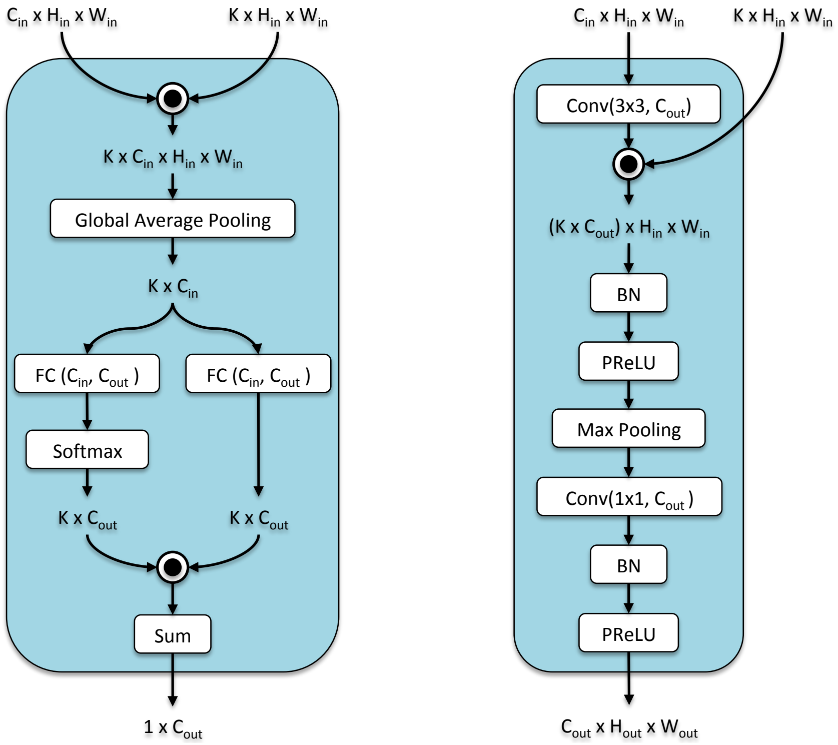

C_{in} x H_{in} x W_{in}
K x H_{in} x W_{in}
K x C_{in} x H_{in} x W_{in}
Global Average Pooling
K x C_{in}
FC (C_{in}, C_{out})
FC (C_{in}, C_{out})
Softmax
K x C_{out}
K x C_{out}
Sum
1 x C_{out}
C_{in} x H_{in} x W_{in}
K x H_{in} x W_{in}
Conv(3x3, C_{out})
(K x C_{out}) x H_{in} x W_{in}
BN
PReLU
Max Pooling
Conv(1x1, C_{out})
BN
PReLU
C_{out} x H_{out} x W_{out}

Figure 4.1: Left: Semantic segmentation-based Pooling (SSP). Right: Semantic segmentation-based Gating (SSG). K indicates the number of semantic regions and C_{out} in SSP equals the number of labels in the main task. We assume that the output tensor of activations from the previous layer to either SSP or SSG is of shape $C_{in} \times H_{in} \times W_{in}$ where C_{in}, H_{in} and W_{in}, respectively represent the number of channels, height and width of the activations.

4.1.2 SSG: Semantic Segmentation-based Gating

Max pooling is used to compress the visual information in the activation maps of the convolution layers. Its efficacy has been proven in many computer vision tasks, such as image classification and object detection. However, attribute prediction is inherently different from image classification. In image classification, we want to aggregate the visual information across the entire spatial domain to come up with a single label for the image. In contrast, many attributes are inherently localized to specific image regions. Consequently, aggregating activations that reside in the “hair" region with the ones that correspond to “mouth”, would confuse the model in detecting “smiling" and “wavy hair" attributes. We propose SSG to cope with this challenge.

Figure 4.1 (Right), shows our proposed SSG architecture. To gate the output activations of the convolution layer, we broadcast element-wise multiplication for each of the semantic regions with the entire activation maps. This generates multiple copies of the activations that are masked differently. In other words, such mechanism spatially decomposes the activations into copies, where high values cannot simultaneously occur in two semantically different regions. For example, gating with the semantic mask that corresponds to the “mouth” region, would suppress the activations falling outside its area while preserving those that reside inside it. However, the area which a semantic region occupies varies from one image to another.

We observed that, directly applying the output of the semantic segmentation network results in instabilities in the middle of the network. To alleviate this, prior to the gating procedure, we normalize the semantic masks such that the values of each channel sums up to 1. We then gate the activations right after the convolution and before the batch normalization [1]. This is very important since the batch normalization [1] enforces a normal distribution on the output of the gating procedure. Then, we can apply max pooling on these gated activation maps. Since, given a channel, activations can only occur within a single semantic region, max pooling operation cannot blend activation values that reside in different semantic regions. We later restore the number of channels using a 1×1 convolution. It is worth noting that SSG can potentially mimic the standard max pooling by learning a sparse set of weights for the 1×1 convolution. In a nutshell, semantic segmentation-based gating allows us to process the activations of convolution layers in a per-semantic region fashion while it also learns how to blend the pooled values back in.

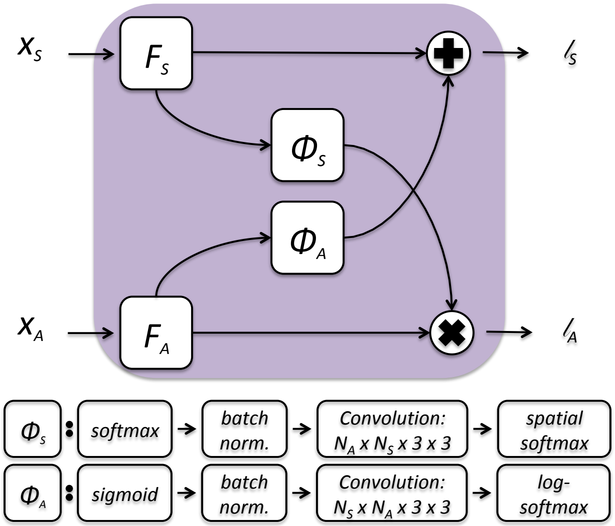

x_S
F_S
Φ_S
Φ_A
x_A
F_A
l_S
l_A
Φ_S : softmax → batch norm. → Convolution: N_A x N_S x 3 x 3 → spatial softmax
Φ_A : sigmoid → batch norm. → Convolution: N_S x N_A x 3 x 3 → log-softmax

Figure 4.2: Architecture of the Symbiotic Augmentation (SA). The embedding layers, Φ_S and Φ_A, respectively utilize the output of semantic segmentation and attribute prediction classifiers, to augment the other task. Details of their components are shown in the bottom. F_S and F_A, are the corresponding classifiers with N_S and N_A number of output labels. Similarly, l_S and l_A indicate the loss functions of semantic segmentation and attribute prediction tasks respectively. For self-expressiveness, we have used different notations here than in Figure 4.1, but in fact N_S and N_A, respectively equal K and C_{out} of SSP in Figure 4.1.

4.1.3 A Simple Unified View to SSP and SSG

In both SSP and SSG architectures, we use the output of semantic segmentation to create K (referring to the number of semantic regions) copies of the activations. Each copy, assuming semantic parsing outputs are perfect, preserves the activation values residing in one semantic region while suppressing those that are outside that. Hence, both SSP and SSG should maintain K times the size of activation maps in the memory. As K value grows, this can certainly become problematic due to limited GPU memory budget.

A simple workaround for this is to learn the masks per channel. Specifically, instead of masking each and every channels of the previous convolution activations by *all* the K semantic probability maps, we learn one mask per channel (ref. Φ_S in Figure 4.2). This can be simply implemented via a 1×1 convolution on the top of semantic segmentation probability maps.

However, in practice, we observed that larger kernels can result in slight performance gain. Similar to SSG, the output logits of the semantic segmentation classifier must be normalized, via batch normalization, prior to being passed to the embedding convolution layer. The output of the embedding should also be spatially normalized (ref. last building block of Φ_S in Figure 4.2). Such embedding allows the model to either pick one or combine (weighted superposition) multiple semantic maps, in order to generate proper mask for each channel. We initialize the convolution kernels ($N_A \times N_S \times 3 \times 3$) of the embedding layers with zeros and no bias. This is inspired by the idea that each channel should start by using all the semantic regions equally. However, through training, it has the freedom to change towards combining only a selected number of regions. We later visualize how the learned convolution kernels of Φ_S look like in Figures 4.9 and 4.8(a).

We now go one step further as the same idea can be used when we reverse the roles of tasks. In particular, we can use the output of attribute prediction to guide the semantic segmentation task. We refer to this joint semantic augmenting model, illustrated in Figure 4.2, as Symbiotic Augmentation (SA). The architecture of the embedding module in this case, Φ_A, is the same as Φ_S except the normalization operations are done differently. Figure 4.2 shows that in Symbiotic Augmentation, each task augments the other task's representation, through its corresponding output logits, while simultaneously being trained in an end-to-end fashion. This is different than SSP and SSG, where only a pre-trained semantic segmentation model, while frozen at deployment, augments attribute prediction task. Note that, in addition to a lower memory footprint, this approach allows us to simplify the SSP by unifying the recognition and localization branches. That is because the learned masks can properly weigh each channel and the order of consecutive linear operations (matrix multiplication through fully connected layer and scaling through weights of localization branch) is interchangeable.

Table 4.1: Configuration of the Semantic Segmentation Network. For all the convolution and deconvolution layers, kernel size and stride value are respectively set to 3 and 1. We compute the semantic segmentation loss at four different scales and aggregate them prior to the backpropagation. To prevent confusion, we are not showing the side loss layers, namely Deconv_{43}, Deconv_{33} and Deconv_{23}.

Layer	Operations	Output Size
Conv_{11}	Conv, BN, PReLU	$64 \times 218 \times 178$
Conv_{12}	Conv, BN, PReLU	$64 \times 218 \times 178$
MaxPool_{1}	Max Pooling	$64 \times 109 \times 89$
Conv_{21}	Conv, BN, PReLU	$128 \times 109 \times 89$
Conv_{22}	Conv, BN, PReLU	$128 \times 109 \times 89$
MaxPool_{2}	Max Pooling	$128 \times 55 \times 45$
Conv_{31}	Conv, BN, PReLU	$256 \times 55 \times 45$
Conv_{32}	Conv, BN, PReLU	$256 \times 55 \times 45$
MaxPool_{3}	Max Pooling	$256 \times 28 \times 23$
Conv_{41}	Conv, BN, PReLU	$512 \times 28 \times 23$
Conv_{42}	Conv, BN, PReLU	$512 \times 28 \times 23$
Deconv_{41}	Deconv, BN, PReLU	$512 \times 28 \times 23$
Deconv_{42}	Deconv, BN, PReLU	$512 \times 28 \times 23$
UpSample_{3}	Up Sampling	$512 \times 55 \times 45$
Deconv_{31}	Deconv, BN, PReLU	$256 \times 55 \times 45$
Deconv_{32}	Deconv, BN, PReLU	$256 \times 55 \times 45$
UpSample_{2}	Up Sampling	$256 \times 109 \times 89$
Deconv_{21}	Deconv, BN, PReLU	$128 \times 109 \times 89$
Deconv_{22}	Deconv, BN, PReLU	$128 \times 109 \times 89$
UpSample_{1}	Up Sampling	$128 \times 218 \times 178$
Deconv_{11}	Deconv, BN, PReLU	$64 \times 218 \times 178$
Deconv_{12}	Deconv, BN, PReLU	$64 \times 218 \times 178$
Deconv_{13}	Deconv, BN, PReLU	$7 \times 218 \times 178$

4.1.4 Semantic Segmentation Network

We previously explained the rationale behind employing semantic segmentation to improve attribute prediction. Our design for semantic segmentation network follows an encoder-decoder approach, similar in concept to the deconvolution network proposed in [25]. However, considering limited number of auxiliary data, we have made different design decisions to reduce the complexity of the model while preserving its capabilities. The encoder consists of 8 convolution layers in blocks of 2, separated with 3 max pooling layers. This is much smaller than 13 layers used in the deconvolution network [25]. At the end of the encoder part, rather than collapsing the spatial resolution as in [25], we maintain it at the scale of one-eighth of the input size.

The decoder is a mirrored version of the encoder replacing convolution layers with deconvolution and max pooling layers with up sampling. Unlike [25] that uses switch variables to store the max pooling locations, we simply up sample the activation maps (repetition with *i.e* nearest neighbor interpolation). We increase (decrease) the number of convolution (deconvolution) filters by a factor of 2 after each max pooling (up sampling), starting from 64 (512) filters as we proceed along the encoder (decoder) path. Every convolution and deconvolution layer is followed by Batch Normalization [1] and PReLU [123] as the non-linearity.

To cope with the challenge of relatively small number of training data, we propagate the semantic segmentation loss at different depths along the decoder path. That is, before each up sampling layer, we compute the loss by predicting the semantic segmentation maps at different scales. We then aggregate these losses with equal weights prior to backpropagation. Finally, while [25] employs VGG-16 [121] weights to initialize the encoder, we train our network from scratch. These design decisions allow us to successfully train the semantic segmentation network with limited number of training data. Detailed configuration of the semantic segmentation network is shown in Table 4.1. In our experiments for SSP and SSG, we use the above semantic segmentation network to generate localization cues in order to improve facial attribute prediction.

4.1.5 Basic Attribute Prediction Network

Our basic attribute prediction model is a fully convolutional neural network, similar in configuration to the encoder part of the semantic segmentation network, detailed in Section 4.1.4. However, we further augment it with four convolution layers. The first two are of size 512 while each of the following ones consists of 1024 channels. Every convolution layer is followed by the Batch Normalization [1] and PReLU [123]. We reduce the size of the activation maps using max pooling when transitioning from 512 to 1024 filters. At the end of the pipeline, we aggregate the activations of the last convolution layer using global average pooling [24] to generate 1024-D vector representations. These vectors will subsequently be passed to the classifier. We train the network using sigmoid cross entropy loss. This architecture is the convolutional backbone in all of our SSP and SSG experiment for facial attribute prediction. It also serves as the average pooling baseline (ref. Section 3.2).

We use AdaGrad [124] with mini-batches of size 32 to train the attribute prediction models from scratch. The learning rate and weight decay are respectively set to 0.001 and 0.0005. We follow the same setting for training semantic segmentation network detailed in Section 4.1.4. At training, we perform data augmentation by randomly flipping (horizontally) the input images. For SSP experiments, we resize the output of the semantic segmentation network at Deconv_{23} layer to 14×12 (resolution of the final convolution layer). To do so, we use max and average pooling operations. Since max pooling increases the spatial support of the region, we use it for the masks associated with eyes, eyebrows, nose and mouth. This helps us to capture some context as well. We use average pooling for the remaining regions. We train the attribute prediction and semantic segmentation networks for respectively 40K and 75K iterations. Next, we detail the backbone architecture that has been used in Symbiotic Augmentation(SA). Unlike SSP and SSG experiments, for which we pretrain the semantic segmentation network and then freeze and integrate it in attribute prediction, here is a single architecture which multi-tasks both semantic segmentation and attribute prediction and does not require any pretraining.

4.1.6 Backbone Architecture for Symbiotic Augmentation(SA)

We use Inception-V3 [26] as the convolutional backbone of Symbiotic Augmentation (SA), for both semantic segmentation and attribute prediction models. Its architecture is 48 layers deep and uses global average pooling instead of fully-connected layers which allows operating on arbitrary input image sizes. Inception-V3 [26] has a total output stride of 32. However, to maintain low computation cost and memory utilization, the size of activation maps quickly reduces by a factor of 8 in only first seven layers, referred to as STEM [26] in Figure 4.3. This is done by one convolution and two max pooling layers that operate with the stride of 2. The network follows by three blocks of Inception layers separated by two grid reduction modules. Spatial resolution of the activations remains intact within the Inception blocks, while grid reduction modules halve the activation size and increase the number of channels. For more details on the Inception-V3 architecture, readers are encouraged to refer to [26].

Symbiotic Augmentation (SA) experiments use a single architecture to simultaneously learn semantic parsing and attribute prediction tasks. This is different from SSP and SSG experiments in which semantic segmentation model (ref. Section 4.1.4) was pretrained and then deployed (weights were frozen) into attribute prediction pipeline (ref. Section 4.1.5). Specifically, we share the weights of the Inception-V3 [26] while training with a mixed minibatch that is comprised of equal instances associated to attribute prediction and semantic segmentation tasks.

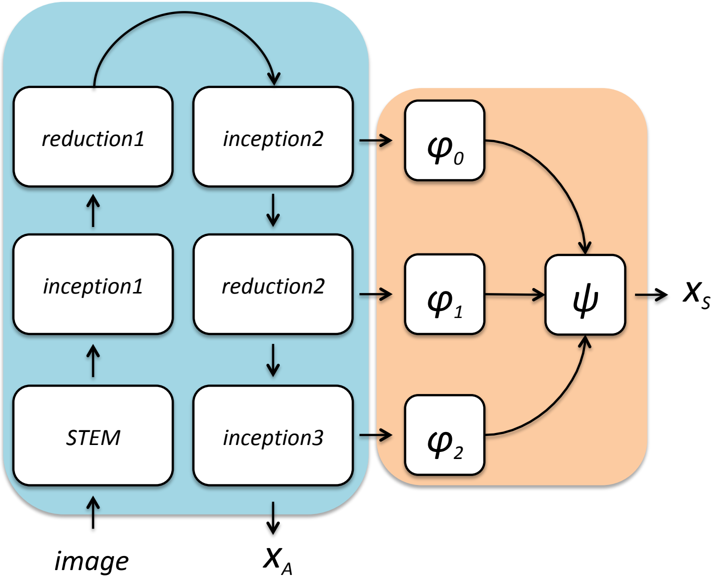


Figure 4.3: Backbone architecture used in the Symbiotic Augmentation (SA) experiments. For attribute prediction, we use the final representation obtained at the end of the Inception-V3 architecture. It is indicated as X_A. For semantic segmentation, we first concatenate final activations of the backbone with two intermediate representations, but only after scaling (with learnable parameters) them using φ_0, φ_1 and φ_2. Then, we reduce the dimensionality of representation to 2048 using a 1×1 convolution, followed by batch normalization and ReLU. Ψ represents the concatenation and dimensionality reduction operations. X_S will be passed to the semantic segmentation classifier.

Figure 4.3 illustrates how we obtain feature representations for both tasks using a single architecture. Note that each element in the minibatch has only one type of annotations, either attribute or semantic segmentation labels. Hence, when X_A and X_S are passed to the Symbiotic Augmentation (SA), shown in Figure 4.2, depending on the annotation type, either l_S or l_A are calculated.

4.2 Experiments

4.2.1 Datasets and Evaluation Measures

We evaluate our proposed attribute prediction models on multiple benchmarks. Specifically, we use CelebA and LFWA [10] for facial attributes, while benchmarking on WIDER Attribute [11] and Berkeley Attributes of People [12] for person attribute prediction. Liu *et al.* [10] have used classification accuracy/error as the evaluation measure on CelebA and LFWA. However, we believe that due to significant imbalance between the numbers of positive and negatives instances per attribute, such measure cannot appropriately evaluate the quality of different methods. Similar point has been raised by [78, 9, 79] as well. Therefore, in addition to the classification error, we also report the average precision (AP) of the prediction scores. Following the literature, we solely report AP for WIDER Attribute [11] and Berkeley Attributes of People [12]. Since attribute benchmarks do not come with pixel-level labels, we train our semantic segmentation model on auxiliary datasets. For experiments corresponding to facial attributes, we use Helen Face [3] along with segment label annotations supplemented by [4]. For person attribute prediction experiments, we train the semantic parsing model on Look into Person (LIP) [23] dataset. We use the standard data split of each corresponding dataset.

CelebA [10] consists of 202,599 images partitioned into training, validation and test splits with approximately 162K, 20K and 20K images in the respective splits. There are a total of 10K identities (20 images per identity) with no identity overlap between evaluation splits. However, we do not use identity annotations. Images are annotated with 40 facial attributes such as, “wavy hair", “mouth slightly open", and “big lips". In addition to the original images, CelebA provides a set of pre-cropped images. We report our results on both of these image sets.

LFWA [10] has a total of 13,143 images of 5,749 identities with pre-defined train and test splits, which divide the entire dataset into two approximately equal partitions. Each image is annotated with the same 40 attributes used in CelebA[10].

WIDER Attribute [11] is collected from 13,789 WIDER images [125], containing usually many people in each image with huge human variations. Each person in these images is then annotated with a bounding box and 14 distinct human attributes such as “longhair”, “sunglasses”, “hat”, “skirt”, and “facemask”. This results in a total of 57,524 boxes. Out of 13,789 images, WIDER Attribute [11] is split into 5,509 training, 1,362 validation and 6,918 test images. There are 30 scene-level labels that each image is annotated with. However, we do not use them and solely train and evaluate on bounding boxes of people. We evaluate on the 29,179 bounding boxes from testing images, after training on 28,345 person boxes extracted from aggregation of training and validation images. Unlike CelebA and LFWA [10], missing attributes are allowed in WIDER Attribute [11] dataset.

Berkeley Attributes of People [12] contains 4,013 training and 4,022 test instances. The example images are centered at the person and labeled with 9 attributes namely, “is male”, “has long hair”, “has glasses”, “has hat”, “has tshirt”, “has long sleeves”, “has shorts”, “has jeans”, “has long pants”. Similar to the WIDER Attribute [11], here unspecified attributes are also allowed.

Helen Face [3] consists of 2,330 images with highly accurate and consistent annotations of the primary facial components. Smith *et. al* [4] have supplemented Helen Face [3] with 11 segment label [1] annotations per image. Images are divided into splits of 2000, 230 and 100, respectively for training, validation and test. Figure 4.4 illustrates a few instances of the input images along with their corresponding segment label annotations. We train our semantic segmentation model on the aggregation of training and validation splits and evaluate on the test split.

[1] “background”, “face skin” (excluding ears and neck), “left eyebrow”, “right eyebrow”, “left eye”, “right eye”, “nose”, “upper lip”, “inner mouth”, “lower lip” and “hair”

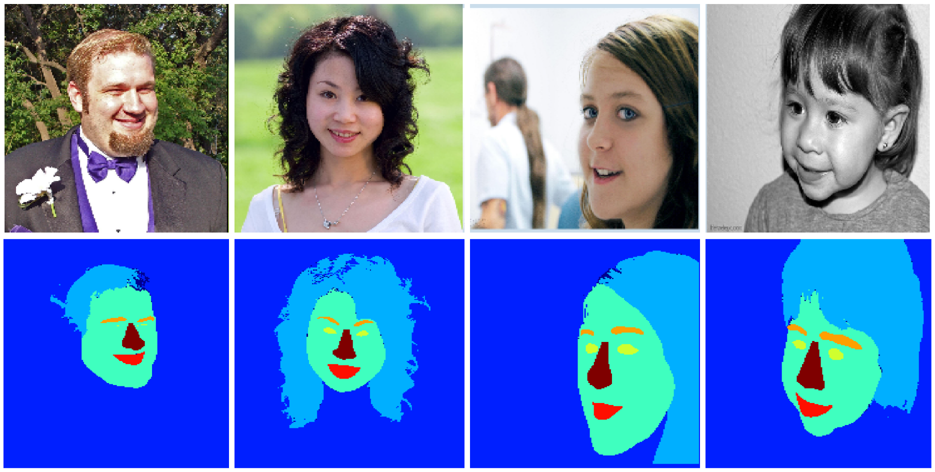


Figure 4.4: Examples of the Helen face dataset [3] supplemented with segment label annotations [4] and then grouped into 7 semantic classes. In bottom row, colors indicate different class labels.

LIP [23] consists of ~30,000 and 10,000 images respectively for train and validation. Each images is annotated with 20 semantic labels[2].

4.2.2 Evaluation of Facial Attribute Prediction

For all the numbers reported here, we want to point out that FaceTracer [64] and PANDA [75] use groundtruth landmark points to attain face parts. Wang *et al.* [126] use 5 million auxiliary image pairs, collected by the authors, to pre-train their model. Wang *et al.* [126] also use state-of-the-art face detection and alignment to extract the face region from CelebA and LFWA images. *However, we train all our models with only attribute and auxiliary face/human parsing labels*. We compare our proposed method with the existing state-of-the-art attribute prediction techniques on the CelebA [10]. To prevent any confusion and have a fair comparison, Table 4.2 reports the perfor-

[2] "Background", "Hat", "Hair", "Glove", "Sunglasses", "Upper-clothes", "Dress", "Coat", "Socks", "Pants", "Jumpsuits", "Scarf", "Skirt", "Face", "Right-arm", "Left-arm", "Right-leg", "Left-leg", "Right-shoe" and "Left-shoe"

mances in two separate columns distinguishing the experiments that are conducted on the original image set from those where the pre-cropped image set have been used. Experimental results indicate that under different settings and evaluation protocols, our proposed semantic segmentation-based pooling and gating mechanisms can be effectively used to boost the facial attribute prediction performance. That is particularly important given that our global average pooling baselines already beat almost all the existing state-of-the-art methods. To see if SSP and SSG are complementary to each other, we also report their combination where the corresponding predictions are simply averaged. We observe that such process further boosts the performance. To investigate the importance of aggregating features within the semantic regions, we replace the global average pooling in our basic model with the spatial pyramid pooling layer [127]. We use a pyramid of two levels and refer to this baseline as SPPNet*. While aggregating the output activations in different locations, SPPNet* does not align its pooling regions according to the semantic context that appears in the image. This is in direct contrast with the intuition behind our proposed methods. Experimental results shown in Table 4.2 confirm that simply pooling the output activations at multiple locations is not sufficient. In fact, it results in a lower performance than global average pooling. This verifies that the improvement obtained by our proposed models is due to their content aware pooling/gating mechanisms.

Naive Approach A naive alternative approach is to consider the segmentation maps as additional input channels. To evaluate its effectiveness, we feed the average pooling basic model with 10 input channels, 3 for RGB colors and 7 for different semantic segmentation maps. The input is normalized using Batch Normalization [1]. We train the network using the same setting as other aforementioned models. Our experimental results indicate that such naive approach cannot leverage the localization cues as good as our proposed methods.

Table 4.2: Attribute prediction performance evaluated by the classification error, average precision and balanced classification accuracy [9] on the CelebA [10] original and pre-cropped image sets.

Classification Error(%)

Method	Original	Pre-cropped
FaceTracer [64]	18.88	–
PANDA [75]	15.00	–
Liu *et al.* [10]	12.70	–
Wang *et al.* [126]	12.00	–
Zhong *et al.* [128]	10.20	–
Rudd *et al.* [78]: Separate	–	9.78
Rudd *et al.* [78]: MOON	–	9.06
AFFAIR [80]	8.55	–
SPPNet*	–	9.49
Naive Approach	9.62	9.13
BBox	–	8.76
Avg. Pooling	9.83	9.14
SSG	9.13	8.38
SSP	8.98	8.33
SSP + SSG	8.84	**8.20**
Inception-V3: baseline	8.68	–
Symbiotic Augmentation (SA)	**8.53**	–

Average Precision(%)

Method	Original	Pre-cropped
AFFAIR [80]	79.63	–
SPPNet*	–	77.69
Naive Approach	76.29	79.74
BBox	–	79.95
Avg. Pooling	77.16	79.74
SSG	77.46	80.55
SSP	78.01	81.02
SSP + SSG	78.74	**81.45**
Inception-V3: baseline	79.28	–
Symbiotic Augmentation (SA)	**80.10**	–

Balanced Accuracy(%) [9]

Method	Original	Pre-cropped
Huang *et al.* [9]	–	84.00
CRL(C) [79]	–	85.00
CRL(I) [79]	–	86.00
Avg. Pooling	–	86.73
SSG	–	87.82
SSP	–	**88.24**

Table 4.3: Attribute prediction performance evaluated by the classification error and the average precision (AP) on LFWA [10] dataset.

Method	Classification Error(%)	AP(%)
FaceTracer [64]	26.00	–
PANDA [75]	19.00	–
Liu *et al.* [10]	16.00	–
Zhong *et al.* [128]	14.10	–
Wang *et al.* [126]	13.00	–
AFFAIR [80]	13.87	83.01
Avg. Pooling	14.73	82.69
SSG	13.87	83.49
SSP	13.20	84.53
SSP + SSG	**12.87**	**85.28**

Table 4.2 shows that at best, the naive approach is on par with the average pooling basic model. We emphasize that feeding semantic segmentation maps along with RGB color channels to a convolutional network results in blending the two modalities in an *additive* fashion. Instead, our proposed mechanisms take a *multiplicative* approach by masking the activations using the semantic segmentation probability maps.

Semantic Masks vs. Bounding Boxes To analyze the necessity of semantic segmentation, we generate a baseline, namely BBox, which is similar to SSP. However, we replace the semantic masks in SSP with the bounding boxes on the facial landmarks. Note that we use the groundtruth location of the facial landmarks, provided in CelebA dataset [10], to construct the bounding boxes. Hence, to some extent, the performance of BBox is the upper bound of the bounding box experiment. There are 5 facial landmarks including left eye, right eye, nose, left mouth and right mouth. We use boxes with area 20^2 (40^2 gives similar results) and 1:1, 1:2 and 2:1 aspect ratios. Thus, there are a total of 16 regions including the whole image itself. From Table 4.2, we see that our proposed models, regardless of the evaluation measure, outperform the bounding box alternative, suggesting that semantic masks should be favored over the bounding boxes on the facial landmarks.

Balanced Classification Accuracy Given the significant imbalance in the attribute classes, also noted by [9, 78, 79], we suggested using average precision instead of classification accuracy/error to evaluate attribute prediction. Instead, Huang *et al.* [9] and later [79] have adopted balanced accuracy measure. To evaluate our proposed approach in balanced accuracy measure, we fine-tuned our models with the weighted ($\propto$ imbalance level) binary cross entropy loss. From Table 4.2, we observe that under such measure, all the variations of our proposed model outperform both [9] and [79] with large margins.

To better understand the effectiveness of our proposed approach on facial attributes, we also report experimental results on the LFWA dataset [10] in Table 4.3. Here, we observe a similar trend to the one in CelebA, where all the proposed models which exploit localization cues successfully improve the baseline. Specifically, SSP + SSG achieves considerably better performance than the average pooling model with margins of 1.86% in classification accuracy and 2.59% in average precision. Our best model also outperforms all other state-of-the-art methods.

Symbiotic Augmentation (SA) All the results reported so far were using a VGG16-like architecture for attribute prediction and a separate pre-trained encoder-decoder architecture for semantic segmentation [2]. However, in SA-based models, we have unified the two architectures and train simultaneously with two objective functions. Table 4.2 shows that simply using a stronger convolutional backbone like Inception-V3 [26] boosts the performance on CelebA original image set. Furthermore, SA-based model which is built on the top of such backbone, despite heavily sharing all the weight across two tasks, can achieve even better results, outperforming SSP+SSG and current state-of-the-art AFFAIR [80]. However, on LFWA dataset [10], we observed that Inception-V3 [26] baseline performs on par with Avg. Pooling baseline reported in Table 4.3 and SA cannot obtain a meaningful gain over its counter global average pooling baseline. We also tried (not reported here) solely using LFWA training instances, without pre-training on CelebA, and observed that SA was indeed effective. However it was not able to reach the performance of the model initialized with CelebA. Detailed per-attribute results of our top models for both CelebA and LFWA datasets are shown in Table 4.4.

Table 4.4: Detailed per-attribute classification accuracy(%) and average precision(%) results of our proposed models for facial attribute prediction. Note that SSP+SSG* indicates the experiment using pre-cropped images of CelebA.

Method	SSP+SSG	SSP+SSG*	Inception -V3: baseline	Symbiotic Aug. (SA)	SSP+SSG	SSP+SSG	SSP+SSG*	Inception -V3: baseline	Symbiotic Aug. (SA)	SSP+SSG
Dataset	CelebA	CelebA	CelebA	CelebA	LFWA	CelebA	CelebA	CelebA	CelebA	LFWA
	Classification Accuracy(%)					**Average Precision(%)**				
5 o Clock Shadow	94.50	95.07	94.34	94.62	79.72	80.36	83.96	80.42	81.63	83.61
Arched Eyebrows	83.06	84.56	83.88	84.12	83.74	77.98	81.17	78.93	79.64	73.07
Attractive	82.25	83.28	82.21	82.27	80.89	91.14	92.50	91.18	91.36	83.83
Bags Under Eyes	85.42	86.15	85.26	85.60	85.09	67.68	70.05	67.24	67.96	95.19
Bald	98.79	99.02	98.92	98.95	92.76	76.43	84.03	79.11	79.40	71.09
Bangs	95.51	96.23	95.72	95.86	91.82	93.86	95.54	94.16	94.65	82.46
Big Lips	71.67	72.45	71.35	72.16	80.20	62.85	62.97	62.30	63.01	81.83
Big Nose	84.50	85.38	84.77	85.01	84.67	68.62	72.25	69.13	71.43	95.92
Black Hair	90.06	90.63	89.96	90.15	92.81	89.75	90.79	89.55	90.13	77.13
Blond Hair	95.82	96.30	95.90	95.94	97.72	91.45	92.73	91.54	91.67	78.77
Blurry	95.67	96.44	95.65	95.85	87.49	53.61	65.87	53.95	57.03	63.88
Brown Hair	89.25	89.95	88.42	88.46	82.72	76.58	78.97	75.22	75.18	83.76
Bushy Eyebrows	92.36	93.20	92.34	92.50	85.77	76.47	81.00	76.36	76.91	94.45
Chubby	95.61	96.02	95.80	95.94	77.66	56.24	62.54	59.63	62.39	76.48
Double Chin	96.28	96.61	96.23	96.47	81.86	58.42	63.92	58.49	61.86	85.80
Eyeglasses	99.27	99.67	99.51	99.48	92.79	98.43	99.20	98.52	98.49	86.96
Goatee	97.28	97.58	97.41	97.55	84.08	74.89	81.64	79.08	80.86	75.74
Gray Hair	98.22	98.37	98.16	98.30	89.24	77.32	80.49	77.65	79.32	71.69
Heavy Makeup	90.83	92.17	91.03	90.99	95.90	96.26	97.31	96.29	96.30	88.80
High Cheekbones	87.13	88.13	87.09	87.48	89.48	94.94	95.78	94.92	95.23	91.68
Male	97.67	98.51	98.00	98.08	94.42	99.59	99.83	99.69	99.73	99.08
Mouth Slightly Open	92.25	94.19	92.61	92.79	84.29	97.97	98.87	98.10	98.29	88.36
Mustache	96.96	97.01	96.94	97.16	94.01	64.14	67.94	65.45	67.01	86.11
Narrow Eyes	86.68	87.92	86.86	87.17	84.68	52.35	59.31	53.22	55.11	95.22
No Beard	95.66	96.52	95.77	95.74	83.63	99.74	99.82	99.76	99.79	94.98
Oval Face	77.83	76.83	77.15	77.50	77.89	66.25	63.84	65.40	65.75	87.21
Pale Skin	97.08	97.29	96.78	96.69	91.15	67.25	70.65	60.60	60.32	97.77
Pointy Nose	76.50	77.86	77.14	77.45	84.99	60.67	65.93	62.74	63.67	95.69
Receding Hairline	93.31	94.14	93.42	93.81	86.60	60.24	67.80	62.05	63.79	95.57
Rosy Cheeks	94.78	95.39	94.75	94.77	86.28	67.66	72.40	64.33	65.41	74.02
Sideburns	97.70	98.00	97.75	97.82	83.21	82.92	86.78	83.16	85.17	81.54
Smiling	91.92	93.39	92.00	92.45	92.51	97.97	98.62	98.07	98.23	97.00
Straight Hair	83.59	84.46	85.16	85.21	81.58	63.56	66.22	68.82	69.21	83.26
Wavy Hair	84.79	84.62	86.13	85.93	81.22	88.46	88.73	90.15	90.27	87.69
Wearing Earrings	89.99	90.94	90.41	90.56	95.23	83.40	85.71	84.79	85.18	89.11
Wearing Hat	98.78	99.11	99.07	99.07	91.08	92.87	95.89	95.21	95.59	75.11
Wearing Lipstick	93.58	94.56	93.61	93.88	95.19	98.67	99.10	98.70	98.76	90.52
Wearing Necklace	88.72	88.01	89.65	89.57	90.15	59.05	52.89	62.92	62.71	82.38
Wearing Necktie	97.15	97.02	97.17	97.12	83.87	86.81	87.51	87.45	88.31	94.47
Young	87.85	89.01	88.52	88.37	86.95	96.89	97.60	97.13	97.19	74.02
Avg.	91.16	91.80	91.32	91.47	87.13	78.74	81.45	79.28	80.10	85.28

4.2.3 Evaluation of Person Attribute Prediction

Table 4.5 compares our proposed method with the state-of-the-art on WIDER Attribute [11] dataset. We observe that the Inception-V3 [26] baseline, despite being considerably shallower, performs on par with ResNet-101. Symbiotic Augmentation (SA) which employs semantic segmentation yields a $\sim$2% performance gain over our Inception-V3 [26] baseline surpassing [129], the current state-of-the-art. For detailed performance comparison between varieties of ResNet [32] and DenseNet [7] architectures on WIDER Attribute [11] dataset, readers are encouraged to refer to [129]. Table 4.6 compares our proposed method with the state-of-the-art on Berkeley Attributes of People [12] dataset. Note that [11] leverages the context in the image while our method solely operates on the bounding box of each person, yet it still outperforms [11] with 2.6% margin. Similar to WIDER Attribute [11] dataset, here utilizing semantic segmentation through our proposed Symbiotic Augmentation (SA) results in 2% gain in AP over our already very competitive Inception-V3 [26] baseline. Detailed per-attribute results of our models are shown in Table 4.7.

Table 4.5: Attribute prediction performance evaluated by the average precision(%) on WIDER Attribute [11] dataset.

Method	AP(%)
Fast R-CNN [130]	80.00
R*CNN [131]	80.50
Deep Hierarchical Contexts [11]	81.30
VeSPA [132]	82.40
ResNet-101 [133]	85.00
ResNet-SRN-att [133]	85.40
ResNet-SRN [133]	86.20
Sarafianos *et. al.* [129]	86.40
Inception-V3: baseline	85.86
Symbiotic Augmentation (SA)	**87.58**

Table 4.6: Attribute prediction performance evaluated by the average precision(%) on Berkeley Attributes of People [12] dataset.

Method	AP(%)
Fast R-CNN [130]	87.80
R*CNN [131]	89.20
Gkioxari *et al.* [74]	89.50
Deep Hierarchical Contexts [11]	92.20
Inception-V3: baseline	92.87
Symbiotic Augmentation (SA)	**94.80**

4.2.4 Visualizations

Figure 4.5 illustrates per-attribute weights that the localization branch of the SSP has learned in order to combine the detection scores associated with different semantic regions. We observe that attributes such as “Black Hair", “Brown Hair", “Straight Hair" and “Wavy Hair" have strong bias towards the hair region. This matches our expectation. However, attribute "Blond Hair" does not behave similarly. We suspect that it is because the semantic segmentation network does not performs as consistent on light hair colors as it does on the dark ones. Attributes such as "Goatee", "Mouth Slightly Open", "Mustache" and "Smiling" are also showing a large bias towards the mouth region. While these are aligned with our our human knowledge, "Sideburns" and "Wearing Necklace" apparently have incorrect biases. We mentioned that semantic pooling funnels back-propagation via multiple pathways, that are semantically aligned in spatial domain, through the network. Unlike the global average pooling which equally affects a rather large spatial domain, we expect SSP to generate activations that are semantically aligned.

Table 4.7: Detailed per-attribute AP(%) results of our proposed models for person attribute prediction.

WIDER Attribute[11]

	Inception-V3: baseline	Symbiotic Augmentation (SA)
Male	95.60	96.64
Long Hair	86.98	89.25
Sunglasses	70.56	78.31
Hat	92.87	95.04
T-shirt	83.36	84.77
Long Sleeve	96.71	97.64
Formal	83.82	85.38
Shorts	91.96	93.87
Jeans	79.60	81.76
Long Pants	97.18	97.74
Skirt	85.74	87.65
Face Mask	76.51	79.18
Logo	91.07	90.87
Stripe	70.15	68.04
Avg.	85.86	87.58

Berkeley Attributes of People [12]

	Inception-V3: baseline	Symbiotic Augmentation (SA)
Is Male	96.29	96.73
Has Long Hair	93.71	94.41
Has Glasses	79.57	88.41
Has Hat	92.97	96.31
Has T-shirt	86.28	88.15
Has Long sleeves	96.96	98.01
Has Shorts	95.43	95.82
Has Jeans	95.34	95.80
Has Long Pants	99.33	99.55
Avg.	92.87	94.80

To evaluate our hypothesis, in Figure 4.6, we show the activations for the top fifty channels of the last convolution layer. Top row corresponds to our basic network with global average pooling, while the bottom row is generated when we replace global average pooling with SSP. We observe that, activations generated by SSP are clearly more localized than those obtained from the global average pooling.

To better understand how attribute prediction and semantic segmentation models have learned their corresponding tasks, we visualize the embedding convolution layers in Φ_S and Φ_A (ref. Figure 4.2) for simultaneously training of CelebA [10] (original image set) with Helen face [3], and WIDER Attribute [11] with LIP [23]. Figure 4.9 shows how for each facial attribute (vertical axis), network has learned to employ different semantic regions of face (horizontal axis) in order to predict attributes. Note that these weights are learned through back-propagation and are not hard coded, yet they reveal very interesting observations. First, almost all the attributes give "background" the lowest importance, except attribute "Wearing Necklace" which makes sense as neck falls outside the face region and counted as background in Helen face dataset [3]. Second, the learned importance for the majority of attributes are aligned with human expectations. For instance, all the hair-related attributes are inferred with the most attention of the model being paid to the "Hair" region. The same is true for "Big Nose", "Pointy Nose" and "Eyeglasses" as the model learns to focus on the "Nose" region. Figure 4.7 illustrates Φ_A for the reverse problem where attributes are supposed to improve semantic face parsing. Figure 4.8(a) and 4.8(b) show the learned weights of the embedding convolution layer for person attribute prediction and human semantic parsing tasks.

We observe that simultaneously training for attribute prediction and semantic segmentation within Symbiotic Augmentation framework, in addition to the performance gains, provides us with meaningful tools to study how a complex deep neural network infers and relate different semantic labels across multiple tasks.

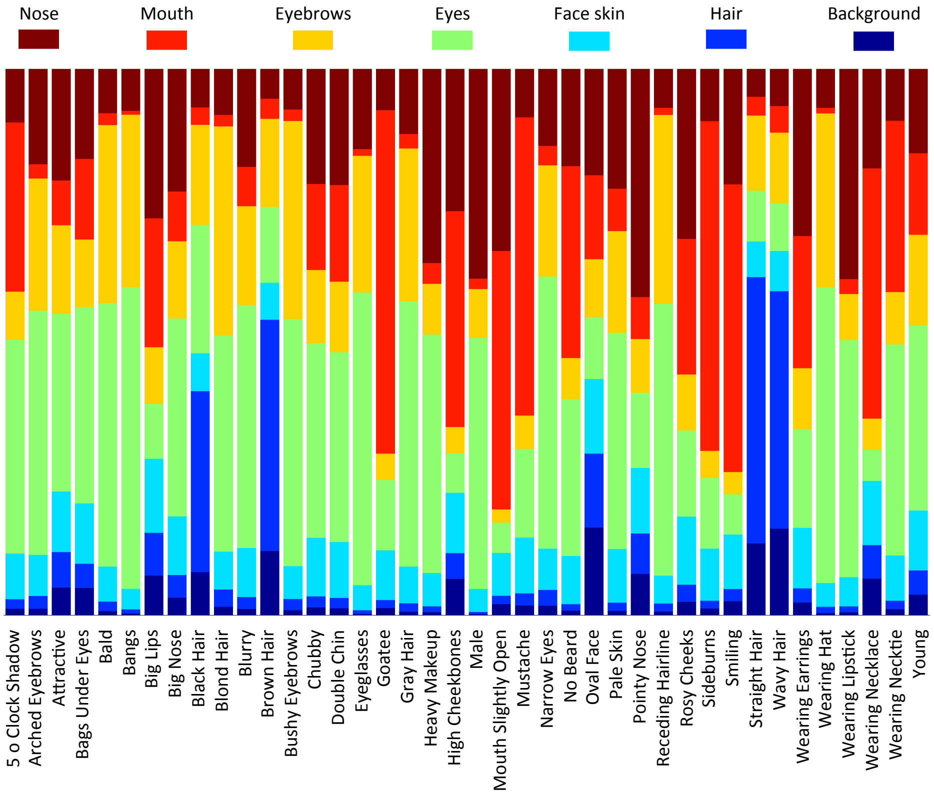


Figure 4.5: Contribution of semantic regions in predicting different attributes as learned by the localization branch of SSP. Values are averaged over multiple random mini-batches of 32 images.

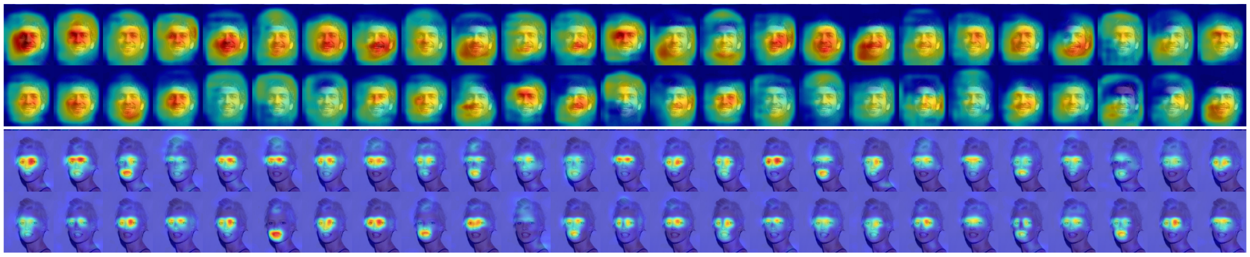


Figure 4.6: Top fifty activation maps of the last convolution layer sorted in descending order w.r.t the average activation values. Top: Basic attribute prediction model using global pooling. Bottom: SSP.

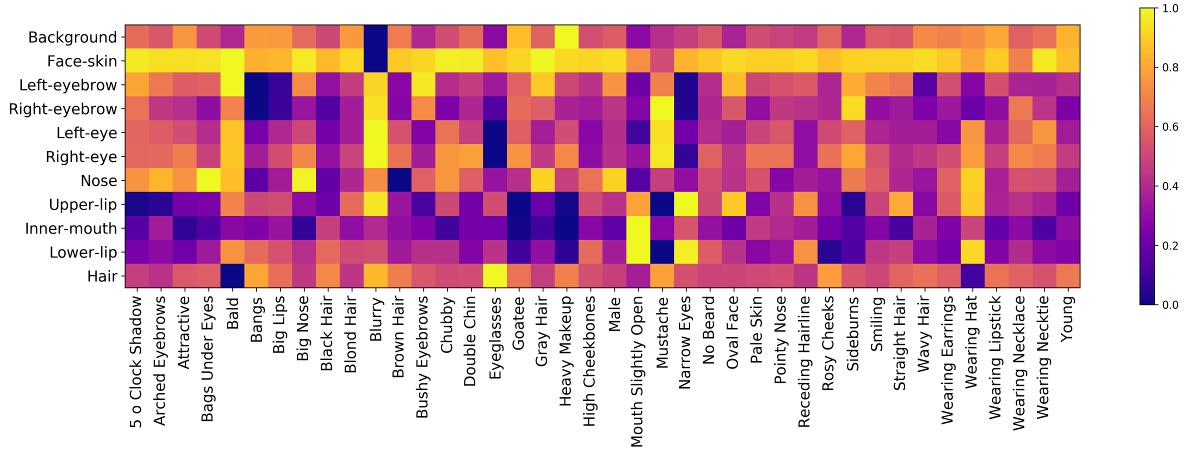
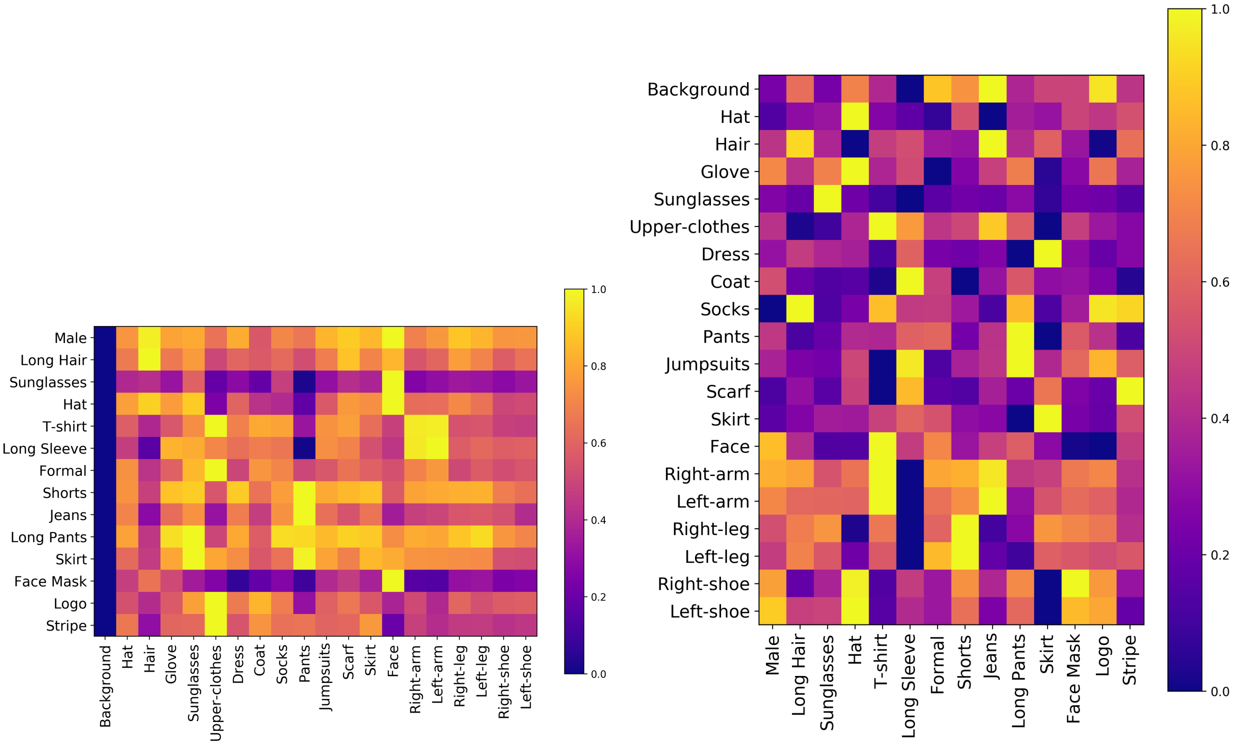


Figure 4.7: Learned weights of Φ_A in Symbiotic Augmentation (SA), trained on CelebA and Helen. Note: 9 values associated with 3×3 kernels are averaged. For better visualization, values in each row are normalized between 0 and 1.

4.2.5 Attribute Prediction for Semantic Segmentation

In this work, we have established how semantic segmentation can be used to improve person-related attribute prediction. What if we reverse the roles. Can attributes improve semantic parsing problem? To evaluate this, we focus on facial attributes and compare the performance of semantic face parsing on Helen face [3]. We consider three scenarios.

First, initializing Inception-V3 [26] backbone with ImageNet [134] pre-trained weights. Second, training a baseline attribute prediction network on CelebA [10] and using the corresponding weights, once training finished, to initialize semantic face parsing network. Third, training facial attribute and semantic face parsing simultaneously through Symbiotic Augmentation (SA) framework. For the sake of simplicity, solely in this experiment, SA only uses the final activations of the CNN backbone instead of concatenating them with intermediate feature maps as shown in Figure 4.3. We observed that upgrading to full SA model boosts mean class accuracy by ∼5% and also achieves similar mean IoU.



(a) Φ_S

(b) Φ_A

Figure 4.8: Learned weights of embedding convolution layers in Symbiotic Augmentation (SA), trained on WIDER and LIP. Note: 9 values associated with 3×3 kernels are averaged. For better visualization, values in each row are normalized between 0 and 1.

Table 4.8 shows that pre-training on image-level facial attribute annotations delivers a large performance gain over ImageNet based initialization. This shows that there exists an *interrelatedness* between attribute prediction and semantic segmentation. Furthermore, it suggests that while collecting annotations for semantic parsing is laborious and expensive, instead one can use relevant image-level attribute annotations to initialize a semantic parsing model. The last row in each block of the Table 4.8 demonstrates how training facial attributes and semantic face parsing jointly, through our proposed Symbiotic Augmentation (SA), can further push the performance boundary with significant margin. Therefore, it is easy to see that when few training instances are available, indeed image-level facial attribute labels can serve as an effective source of weak supervision to improve semantic face parsing task. In fact such *interrelatedness* plays a major role in allowing us to successfully unify semantic segmentation and attribute predictions networks (ref. Section 4.1) without sacrificing the performance. Jointly training on LIP [23] and WIDER Attribute [11], we did not observe meaningful gain in semantic segmentation task on LIP [23]. We hypothesize that, this is due to the fact that LIP [23] itself already has huge (∼30,000 instances) number of training annotations. In order to confirm this, conducting an experiment where only a small portion of LIP [23] training instances are utilized is needed.

Table 4.8: Effect of leveraging image-level attribute supervision for semantic face parsing, evaluated on the test split of Helen face [3][4]. Here, all the models were trained with the input image resolution of 448 × 448.

Intersection over Union(%)												
Method	bkg	face skin	l-eyebrow	r-eyebrow	l-eye	r-eye	nose	u-lip	i-mouth	l-lip	hair	**Avg.**
init: ImageNet	92.97	85.58	46.90	48.33	55.39	55.91	84.24	43.77	59.21	55.19	71.99	63.58
init: CelebA	93.20	86.40	51.31	51.11	56.22	58.81	84.82	49.32	60.01	58.95	73.13	65.75
SA	**94.25**	**88.24**	**59.29**	**58.11**	**62.45**	**67.22**	**87.96**	**51.05**	**69.66**	**70.32**	**75.77**	**71.29**
Class Accuracy(%)												
method	bkg	face skin	l-eyebrow	r-eyebrow	l-eye	r-eye	nose	u-lip	i-mouth	l-lip	hair	**Avg.**
init: ImageNet	96.04	94.21	56.02	60.95	67.61	67.62	90.69	58.25	74.73	66.12	83.36	74.14
init: CelebA	95.96	94.09	63.31	67.71	67.30	69.79	90.06	66.80	75.27	72.83	85.22	77.12
SA	**97.02**	**95.47**	**69.89**	**74.97**	**72.12**	**77.21**	**92.43**	**66.96**	**76.88**	**81.60**	**84.67**	**81.07**

4.3 Summary

Aligned with the trend of part-based attribute prediction methods, we proposed employing semantic segmentation to improve person-related attribute prediction. Specifically, we jointly learn attribute prediction and semantic segmentation in order to mainly transfer localization cues from the latter task to the former. To guide the attention of our attribute prediction model to the regions which different attributes naturally show up, we introduced SSP and SSG. While SSP is used to restrict the aggregation procedure of final activations to regions that are semantically consistent, SSG carries the same notion but applies it to the earlier layers. We then demonstrate that there exist a single unified architecture that can mimic the behavior of SSP and SSG, depending on where in the network architecture it is being used. We evaluated our proposed methods on CelebA, LFWA, WIDER Attribute and Berkeley Attributes of People datasets and achieved state-of-the-art performance. We also showed that attributes can improve semantic segmentation (in case of few training instances) when properly used through our Symbiotic Augmentation (SA) framework. We hope to encourage future research works to invest more in the interrelatedness of these two problems.

In the next Chapter, we first introduce our very own Selfie dataset. Then, we follow through by detailing our approach in analysing the Selfie images using state-of-the-art computer vision techniques.

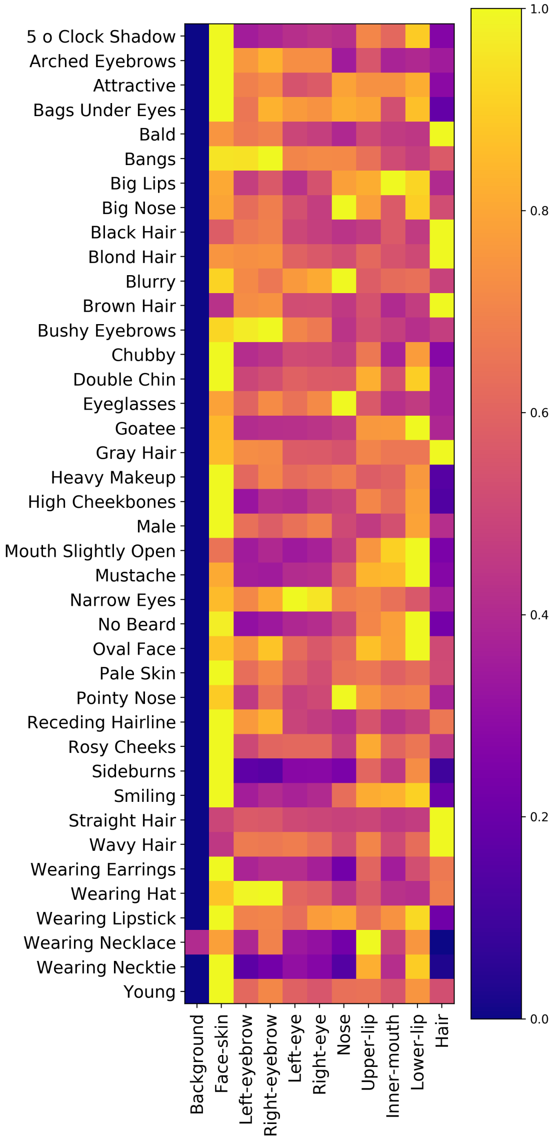


Figure 4.9: Learned weights of Φ_S in Symbiotic Augmentation (SA), trained on CelebA and Helen. Note: 9 values associated with 3×3 kernels are averaged. For better visualization, values in each row are normalized between 0 and 1.

CHAPTER 5: ANALYSIS OF SELFIE IMAGES

The results of this Chapter have been published in the following paper:

Mahdi M. Kalayeh, Misrak Seifu, Wesna LaLanne, Mubarak Shah, *“How to Take a Good Selfie?,”* in Proceedings of the 23rd ACM International Conference on Multimedia, 2015, pp. 923-926.[135]

The massive number of self-portrait images taken and shared on social media is revolutionizing the way people introduce themselves and the circle of their friends to the world. The Big Data nature of Selfies, naturally demands algorithmic approaches to automate understanding and inference from Selfies, as it is nearly impossible to analyze them manually. We show that both textual tags and visual attributes, as computational semantic tools can be effectively utilized to analyze Selfies in scale. To do so, we begin by collecting the first Selfie dataset with more than 46K images and annotate it with 36 visual attributes covering characteristics such as gender, age, race, and hairstyle. We provide attribute prediction of Selfies, using SIFT and HOG, pre-trained AlexNet on ImageNet, and Adjective Noun Pairs (*e.g “smiling boy”*) of SentiBank. In order to assess the impact of different visual concepts or various Instagram filters on the popularity of Selfies, we train ℓ_2-regularized Support Vector Regression (SVR) to estimate the $\log_2$-normalized view counts. We also study how popularity and sentiment of Selfies are correlated.

This Chapter begins with the details of our proposed selfie dataset in Section 5.1. This includes the collection and annotation processes. We then complement our dataset by providing baseline attribute prediction in Section 5.2 using four different, including deep and non-deep feature representations. This chapter continues as we address multiple research questions regarding popularity and sentiment of selfies in Sections 5.3.1, 5.3.2, and 5.3.3, all experimented on our very own proposed selfie dataset.

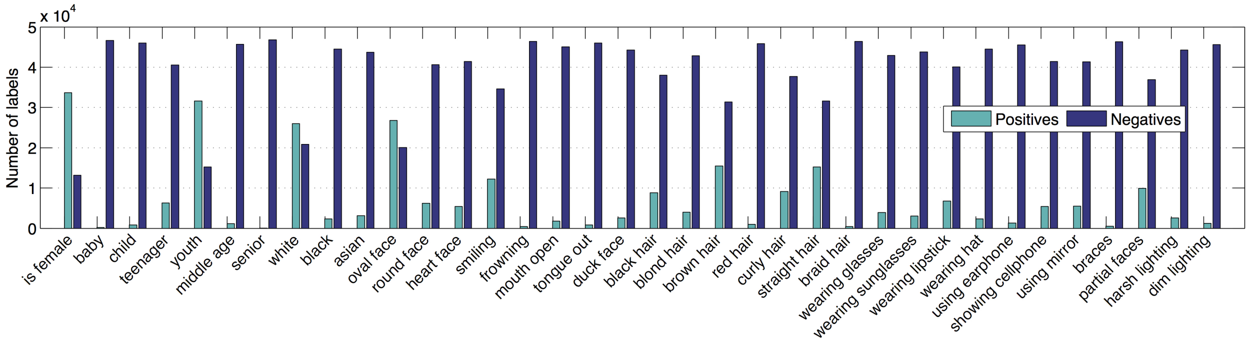


Figure 5.1: Number of labeled positive and negative images in the Selfie dataset for different attributes.

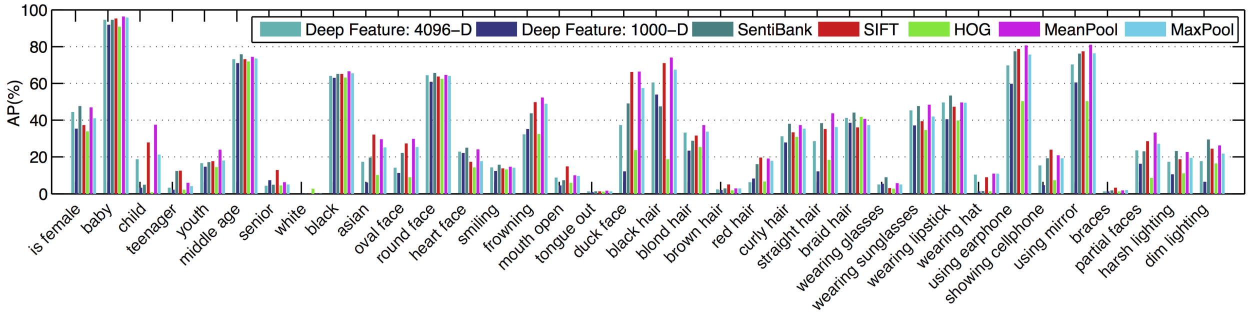


Figure 5.2: Attribute prediction performance of different features on the Selfie dataset.

5.1 Selfie Dataset

Due to the massive number of selfies being uploaded on social media every minute, any study aiming to provide insights about selfies has to be conducted on very large number of images to ensure that it sufficiently captures the variations in the data. To the best of our knowledge, there exists no selfie dataset publicly available for research purposes. Therefore, we collected our own dataset. We downloaded 85,000 images from `selfeed.com`, a real-time update of #*selfie* on Instagram. Despite being tagged with #*selfie*, we found that only 69,710 of those images are actually selfie images. The remaining 15,290 images were either completely irrelevant or general photographs of people. This is an interesting observation, since different social media usually report the number of selfies shared on their environment by counting the images tagged with #*selfie*. However, in our data collection, at least 18% of images tagged with #*selfie* are not, in fact, selfies.

In preparation of the Selfie dataset [135], we made sure to discard any images that do not fall under the definition of *selfie*. Clearly this is a subjective judgment and different annotators may disagree in a few cases whether a photo is a selfie or not. In order to generate a uniform understanding of *selfie* among our annotators, we asked them to annotate a fixed set of images containing about 2,000 images. Then, the images with mixed votes where discussed and disagreement was resolved by clarifying the definition. About 32% of selfie images that we collected were showing multiple faces either in form of group selfies or collages. Excluding multiple-face images yields to a total number of 46,836 single-face selfies. Since we wanted to annotate selfies with attributes such as age, gender, hair color and etc., we were mostly interested in selfies that do not show multiple faces. Figure 1.4 shows some of the collected selfie images. We annotated 46,836 selfie images with 36 different attributes divided into several categories as follows: *Gender:* is female. *Age:* baby, child, teenager, youth, middle age, senior. *Race:* white, black, asian. *Face shape:* oval, round, heart. *Facial gestures:* smiling, frowning, mouth open, tongue out, duck face. *Hair color:* black, blond, brown, red. *Hair shape:* curly, straight, braid. *Accessories:* glasses, sunglasses, lipstick, hat, earphone. *Misc.:* showing cellphone, using mirror, having braces, partial face. *Lighting condition:* harsh, dim. We asked annotators to look into a random subset ($\sim$3,000 images) of the Selfie dataset and create a list of attributes that they 1) frequently observe and 2) can easily detect. These two conditions assure enough positive samples for each attribute and an acceptable performance for the attribute detectors. Figure 5.1 shows the ground truth statistics of the collected Selfie dataset.

5.2 Attribute Prediction

To analyize selfie images outside our dataset, we have to be able to predict different attributes of interest with an acceptable precision. Therefore, we provide a baseline for attribute prediction along with introduction of our Selfie dataset. In our experimental setting, for every attribute, positive instances were randomly divided into 3 folds. We do the same for the negative instances. Training split for each attribute consists of 2 out of 3 folds from both positive and negative instances. The third folds from positive and negative instances, together create the testing split. For feature extraction from images, we use

SIFT We densely extract SIFT descriptors from images at every 3 pixels and at 8 different scales. To encode these descriptors into a single feature vector, we employ VLAD [136] with a codebook size of 256. **HOG** We densely extract HOG descriptors from images with the cell size of 8 pixels. Descriptor encoding is the same as the one used for dense SIFT. **Deep Features** Using deep convolutional neural networks (CNN) [137] trained on ImageNet dataset [138], we extract 4096-D feature vectors from CNN's last fully connected layer. We also use final 1000-D classification layer as another feature representation. We employed the large network of OverFeat [139] to implement CNN. **SentiBank** Authors in [28] have trained 2,089 visual concept detectors based on a list of Adjective Noun Pairs (ANP). These ANPs are like *smiling boy*, *lovely dress*, *scary face* and etc. Using SentiBank, we generate a 2089-D vector for each image where every dimension is the detection score of its corresponding ANP concept detector.

We employed *one-vs-all* support vector machine (SVM) with linear kernel to train attribute detectors. The performance of detectors is measured via average precision (AP). Figure 5.2 shows the performance of different features for the task of attribute detection on the Selfie dataset. Deep features extracted from last fully connected layer of CNN perform better (29.51% vs 24.03% meanAP) than the 1000-D features from CNN's classification layer. Features obtained by applying SentiBank ANP concept detectors achieve the best performance with 31.97% meanAP among three different mid-level features. We expected to observe this since ANPs are very diverse and a very large portion of them are relevant to human attributes. Using SIFT and HOG (low-level features) with VLAD encoding results in 33.76% and 22.95% meanAP, respectively. We also fused 4096-D deep features, SentiBank features and SIFT via mean and max pooling their detection scores (late fusion). While max pooling was not helpful (32.49% meanAP), mean pooling further improved the attribute prediction baseline to 35.79% meanAP.

5.3 Experiments

5.3.1 What Makes a Selfie Popular?

In this section, we attempt to answer the second question that we proposed in the beginning of this paper. How does the appearance of certain objects or particular concepts influence the popularity of selfie images? We discuss the effect of attributes in section 5.3.3. Here we study object categories of ImageNet [138] and concepts associated with ANPs in SentiBank [28]. To evaluate the correlation of each object/concept with popularity of selfie images, we use 1000-D output of [139], pre-trained on ImageNet dataset, and 2089-D output of SentiBank ANP concept detectors as mid-level features. We then train an L_2-regularized support vector regression (SVR) to predict the popularity score of the images. Regression coefficients corresponding to different objects/concepts indicate their correlation to the popularity. We use object/concept detector responses with at least 0.5 confidence. This assures that we only consider images in which a particular object/concept is confidently detected. We observed that among ImageNet object categories: *maillot, lab coat, jersey, fur coat, brassiere, wig, abaya, hair spray, suit, sunglasses, and lipstick* (in decreasing order) are the most relevant objects to the popularity of selfies. Among different ANPs in SentiBank: *sexy dress, lovely dress, fancy dress, traditional tattoo, smiling baby, shiny hair, sexy girls, cute baby,* *strong legs, stupid hat and happy baby* (in decreasing order) are the most relevant concepts to the popularity score. Figure 5.3 illustrates the normalized regression coefficients (in decreasing order) obtained from training SVR.

Popularity Score Prediction: Observing the performance of different features in attribute prediction, we are also interested to see how they perform in predicting the popularity of selfie images. Thus, we randomly divided the entire Selfie dataset using 3-fold cross validation where 2 folds were used to train an L_2-regularized SVR and we tested on the third fold. We evaluate the quality of different features in terms of Spearman's rank correlation between the predicted and the actual popularity (generated by [27]). Using SIFT, we achieved 0.40 rank correlation. The 4096-D deep features and 2089-D SentiBank features resulted in 0.41 and 0.54 rank correlation, respectively. While max pooling of these features was not helpful (0.49 rank correlation), we observe that mean pooling can further boost the rank correlation to 0.55.

5.3.2 Sentiment-Popularity Correlation

In this section we explore the correlation between the popularity of selfie images and their implied sentiments. Each ANP in SentiBank is associated with a sentiment measure where negative, close to zero and positive numbers depict negative, neutral and positive sentiments, respectively.

Out of 2,089 ANPs, we manually select 126 of them that are relevant to selfie images and their corresponding detectors have acceptable performance (AP$\geq$ 0.3). We compute implied sentiment of a selfie image I as $\mathscr{S}(I) = \sum_i \gamma_i s_i \omega_i^T x$, where γ_i, s_i and ω_i, respectively, represent the AP, sentiment measure and linear detector corresponding to the i^{th} ANP. Given x as the feature representation of image I, $\omega_i^T x$ is the confidence score for which i^{th} ANP concept appears in the image I. We generate the scatter plot of the popularity versus sentiment using our entire dataset.

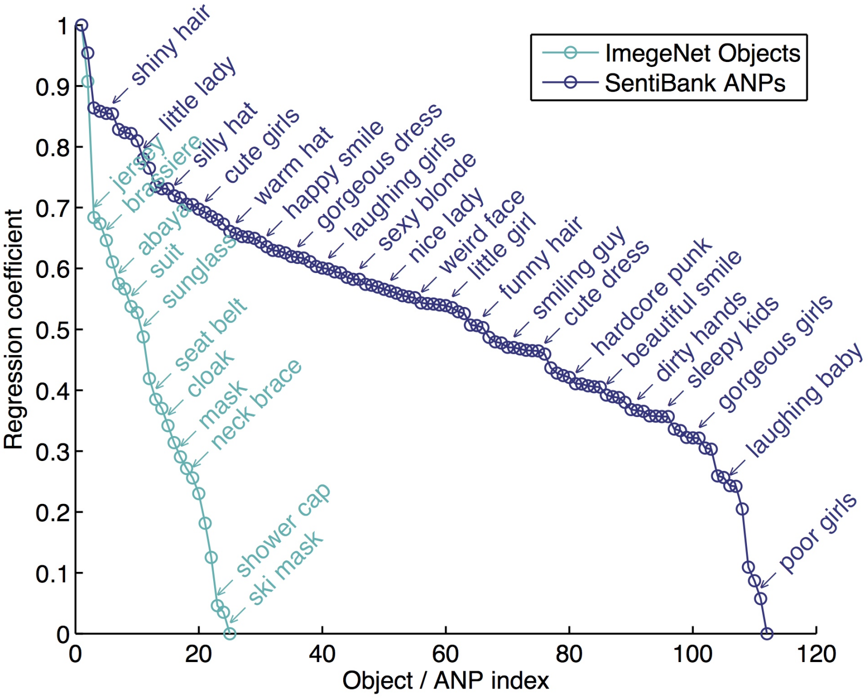


Figure 5.3: Normalized regression coefficients of SVR for popularity score prediction.

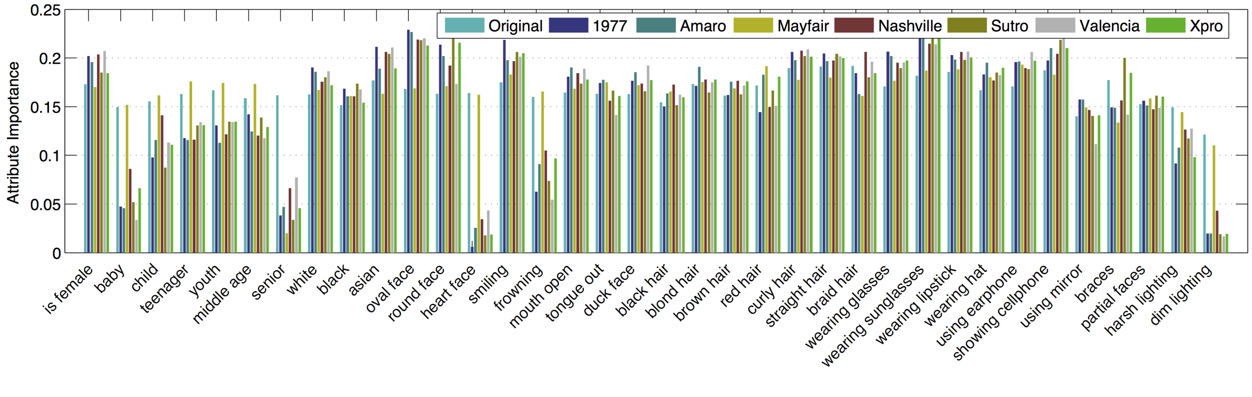


Figure 5.4: Importance of different attributes in predicting popularity, employing different Instagram filters. *Original* indicates no filter is applied.

Figure 5.5 illustrates the scatter plot in which a more positive sentiment, on average, results in a higher popularity. We observe, on average, up to 65% higher popularity comparing the two ends of the sentiment spectrum, shown in color. Another interesting observation is how the range of popularity changes as the sentiment measure increases. From bluish to green/yellow parts of the spectrum, increasing sentiment yields a larger range of popularity score (the blue cone). However, moving toward more reddish parts of the spectrum, the range of the popularity decreases. Therefore, we conclude that while at two ends of the sentiment spectrum there exist a direct correlation between sentiment and popularity, this is not true for the middle of the spectrum. In other words, unless the sentiment is too high or too low, one cannot estimate the popularity based on the sentiment with high precision.

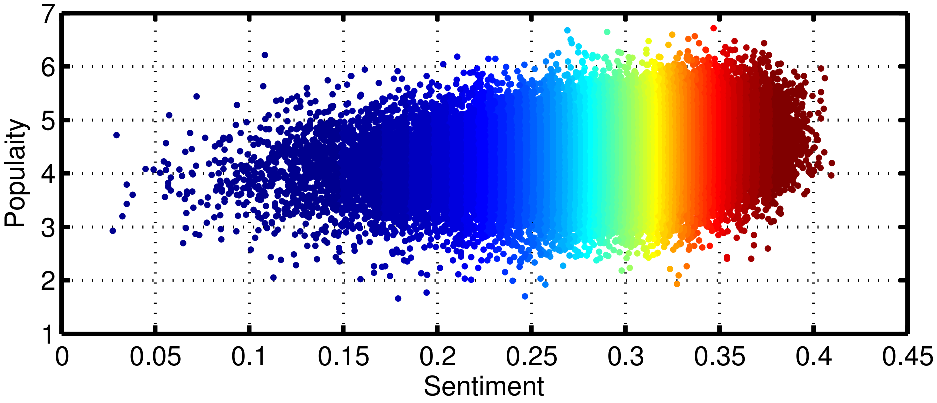


Figure 5.5: Sentiment-popularity scatter plot.

5.3.3 Effect of Post-processing on Popularity

This section addresses the last question that we proposed in the beginning of this paper: How does post-processing, such as applying different Instagram filters, influence the popularity of selfies? We randomly select about 10,000 images from our Selfie dataset and apply 7 different Instagram filters to them. We observe that for a given selfie, applying some filters boosts the popularity while others have a counter-effect. We found that there is no definite ranking of filters in terms of improving the popularity, rather ranking varies from one image to another. Figure 5.4 shows the relevance of different attributes to popularity of selfies when various Instagram filters are applied. Therefore, the choice of the most effective filter varies according to the content of a selfie. We obtain the importance of different attributes using SVR in a similar strategy discussed in section 5.3.1.

5.4 Summary

Interested in the utilizing computer vision to analyze greater social behaviors, in this chapter, we addressed, for the first time, the Selfie phenomenon from a machine learning and computer vision perspective. We conducted a series of experiments addressing four distinct questions about selfie images. Our work shed light on how to take a *good selfie*, a selfie that becomes popular and delivers a positive sentiment. We have collected, to the best of our knowledge, the first Selfie dataset for research purposes with more than 46,000 images, annotated with 36 relevant attributes that describe detail local properties of human. Our work has attracted researchers from other disciplines to extend their studies on the selfie phenomenon to a larger-scale.

Next, we present our mixture normalization, a novel soft piece-wise alternative to batch normalization, that attempts to accelerate training deep convolutional neural networks by disentangling modes of variation in output activations of the intermediate layers.

CHAPTER 6: TRAINING FASTER BY SEPARATING MODES OF VARIATION IN BATCH-NORMALIZED MODELS

The results of this Chapter have been published in the following paper:

Mahdi M. Kalayeh, Mubarak Shah, *"Training Faster by Separating Modes of Variation in Batch-normalized Models,"* in IEEE transactions on Pattern Analysis and Machine Intelligence (PAMI), 2019.[140]

Today, Batch Normalization (BN) [1] is the de facto standard in almost every deep learning architecture. However, despite its wide usage, the literature falls short when it comes to comprehensively studying its characteristics and behavior. We revisit BN from a perspective, orthogonal to what previous extensions to BN have proposed. We show that assuming samples within a mini-batch are from the same probability density function, then BN is identical to the Fisher vector of a Gaussian distribution. That means batch normalizing transform can be explained in terms of kernels that naturally emerge from the probability density function that models the generative process of the underlying data distribution. Specifically, we theoretically demonstrate how BN can be improved by disentangling modes of variation in the underlying distribution of layer outputs.

In this Chapter, we start by showing how the formulation of batch normalization and its extensions is related to the kernels from generative probability models, and more specifically Fisher kernel, in Section 6.1.1. Then, Section 6.1.2 explains why in the context of deep neural networks, due to the non-linearities, distribution of the activations is almost certainly of multiples modes of variation. This is followed by proposing Mixture Normalization, where we employ Gaussian mixture model to identify sub-populations and normalize with respect to not one, but multiple components that comprise the data distribution. We conclude this Chapter in Section 6.2 where we show that our proposed Mixture Normalization, not only effectively accelerates training of different batch-normalized architectures including Inception-V3, DenseNet, and DCGAN, but also achieves better generalization error.

6.1 Methodology

6.1.1 Kernels from Generative Probability Models

So far, we've described variations of batch normalization in an unifying framework and shown what distinguishes them is the set of samples over which certain statistics, specifically mean and standard deviation are computed. In this section, we revisit Fisher kernel, the seminal work of Jaakkola and Haussler [30], in order to demonstrate how the general form of normalization, specified in Equation 2.6, is related to the kernels that naturally arise from the generative probability models.

Fisher Kernel Let $\mathscr{B} = \{x_{1\dots m}\}$, regardless of how the tensor of activations has been indexed (ref. Section 2.4), be a set of m observations $x_i \in \mathscr{X}$, associated to a sample mini-batch, and $\mathscr{P} = \{p_\theta, \theta \in \Theta\}$ be a suitably regular parametric family of distributions, where p_θ models the generative process of samples in $\mathscr{X}$[1]. Based on the theory of information geometry [141], $\mathscr{P}$ defines a Riemanninan manifold M_Θ, with a local metric given by the Fisher Information Matrix

$$F_\theta = \mathbb{E}_{x \sim p_\theta}[G_\theta^{\mathscr{B}} G_\theta^{\mathscr{B}^T}], \tag{6.1}$$

[1] In other words, $\mathscr{X}$ is an unknown generative process which is modeled by p_θ, and we assume that instances in a mini-batch are sampled from it.

where

$$G_\theta^{\mathscr{B}} = \nabla_\theta \log p_\theta(\mathscr{B}), \tag{6.2}$$

the gradient of the log-likelihood of p_θ at $\mathscr{B}$, also known as *Fisher score*, determines the direction of steepest ascent in $\log p_\theta(\mathscr{B})$ along the manifold. In other words, $G_\theta^{\mathscr{B}}$ describes modifications which the current model parameters θ need in order to maximize $\log p_\theta(\mathscr{B})$ [30]. The natural gradient [142] is then defined as

$$\phi_{\mathscr{B}} = F_\theta^{-1} G_\theta^{\mathscr{B}}, \tag{6.3}$$

based on which [30] proposes natural kernel, the inner product between natural gradients relative to the local Riemannian metric:

$$K(\mathscr{B}_j, \mathscr{B}_i) \propto \phi_{\mathscr{B}_j}^T F_\theta \phi_{\mathscr{B}_i} = G_\theta^{\mathscr{B}_j^T} F_\theta^{-1} G_\theta^{\mathscr{B}_i}. \tag{6.4}$$

Jaakkola and Haussler [30] refer to this as *Fisher kernel* and prove that given classification labels as latent variable, Fisher kernel is asymptotically never inferior to the maximum *a posteriori* (MAP) decision rule (ref. Theorem 1 in [30]). Hence, we arrive at a similarity measure which is naturally induced from the probability density function that models the generative process of $\mathscr{X}$ *and* simultaneously improves the discrimination power of the model.

Sanchez *et. al* [143] have beautifully pointed out that since F_θ is positive semi-definite, its inverse has a Cholesky decomposition $F_\theta^{-1} = L_\theta^T L_\theta$, which allows re-writing Equation 6.4 as

$$K(\mathscr{B}_j, \mathscr{B}_i) = \mathscr{G}_\theta^{\mathscr{B}_j^T} \mathscr{G}_\theta^{\mathscr{B}_i}, \tag{6.5}$$

where

$$\mathscr{G}_\theta^{\mathscr{B}} = L_\theta \nabla_\theta \log p_\theta(\mathscr{B}), \tag{6.6}$$

the normalized gradient vector of $\mathscr{B}$, is known as *Fisher vector* [143]. Such explicit transformation $\mathscr{B} \to \mathscr{G}_\theta^{\mathscr{B}}$ enjoys all the interesting characteristics of Fisher kernel [30], while being tailored for linear operations.

Fisher Vector for Gaussian Distribution In order to derive Fisher kernel [30], we noted that p_θ only needs to be of a family of regular parametric distributions. However, we have not yet parameterized it. Let's consider p_θ to be modelled by Gaussian distribution

$$p_\theta(x) = \frac{1}{(2\pi)^{D/2}|\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu) \right\}, \tag{6.7}$$

where $x \in \mathbb{R}^D$ and $\theta = \{\mu, \Sigma\}$ is the set of model parameters. The gradients of the log-likelihood of $p_\theta(x)$ with respect to μ and Σ are then formulated as

$$\begin{aligned}
\nabla_\mu \log p_\theta(x) &= -\tfrac{1}{2}\Big(\tfrac{\partial (x-\mu)^T\Sigma^{-1}(x-\mu)}{\partial \mu}\Big) = \Sigma^{-1}(x-\mu), \\
\nabla_\Sigma \log p_\theta(x) &= -\tfrac{1}{2}\Big(\tfrac{\partial \log(|\Sigma|)}{\partial \Sigma} + \tfrac{\partial (x-\mu)^T\Sigma^{-1}(x-\mu)}{\partial \Sigma}\Big), \\
&= -\tfrac{1}{2}\Big(\Sigma^{-1} - \Sigma^{-1}(x-\mu)(x-\mu)^T\Sigma^{-1}\Big).
\end{aligned} \tag{6.8}$$

If we assume Σ to be diagonal [2] with $\Sigma_{i,i} = \sigma_{i,i}^2$, we can re-write the gradients as

$$\begin{aligned}
\nabla_\mu \log p_\theta(x) &= \frac{1}{\sigma^2}(x-\mu), \\
\nabla_\sigma \log p_\theta(x) &= -\frac{1}{\sigma} + \frac{1}{\sigma^3}(x-\mu)^2.
\end{aligned} \tag{6.9}$$

Substituting Equation 6.9 in Equation 6.2 results in

$$G_\theta^x = \begin{bmatrix} \nabla_\mu \log p_\theta(x) \\ \nabla_\sigma \log p_\theta(x) \end{bmatrix} = \begin{bmatrix} \frac{1}{\sigma^2}(x-\mu) \\ -\frac{1}{\sigma} + \frac{1}{\sigma^3}(x-\mu)^2 \end{bmatrix}. \tag{6.10}$$

[2] Such assumption is motivated by the fact that computing full covariance matrix and its inverse makes both the normalization and the back-propagation calculations very expensive [1].

We then use non-central moments of univariate normal distribution to calculate Equation 6.1 given Equation 6.10, arriving at

$$F_\theta = \begin{bmatrix} \frac{1}{\sigma^2} & 0 \\ 0 & \frac{2}{\sigma^2} \end{bmatrix}, \tag{6.11}$$

the Fisher information matrix of Gaussian distribution. Finally, the Fisher vector for Gaussian distribution (ref. Equation 6.6) can be characterized as

$$\begin{aligned} \mathscr{G}_\theta^x &= \begin{bmatrix} \mathscr{G}_\mu^x \\ \mathscr{G}_\sigma^x \end{bmatrix}, \\ &= \begin{bmatrix} \sigma & 0 \\ 0 & \frac{1}{\sqrt{2}}\sigma \end{bmatrix} \begin{bmatrix} \frac{1}{\sigma^2}(x-\mu) \\ -\frac{1}{\sigma} + \frac{1}{\sigma^3}(x-\mu)^2 \end{bmatrix}, \\ &= \begin{bmatrix} \frac{1}{\sigma}(x-\mu) \\ \frac{1}{\sqrt{2}}\Big(-1 + \frac{(x-\mu)^2}{\sigma^2}\Big) \end{bmatrix}. \end{aligned} \tag{6.12}$$

We observe that the general form of normalization, formulated in Equation 2.6, has in fact emerged in $\mathscr{G}_\mu^x$. Hence, we have shown that indeed batch normalization and its various extensions are closely related to the natural kernel that arises from generative probability model, describing the underlying data distribution [3].

6.1.2 Mixture Normalization

We believe that in the context of deep neural networks, due to non-linear activation functions, distribution of layer outputs are very likely asymmetric. That is, the hypothesis which a Gaus-

[3] Based on our derivations, we "expect" that integrating $\mathscr{G}_\sigma^x$ in addition to $\mathscr{G}_\mu^x$ further improves the normalization. Yet, at the time, we do not have experimental results to support this.

sian distribution can model p_θ is less likely to be valid. Therefore, to properly approximate the probability density function, we propose employing generative *mixture* models. Consequently, instead of computing one set of statistical measures from the entire population, we propose to frame batch normalizing transform on sub-populations which can be identified by disentangling modes of variation.

Intuition Let's consider $x_l \in \mathbb{R}^{N \times C_l \times H_l \times W_l}$ to be the input activation tensor to the l^{th} layer, denoted by ω_l, in a convolutional neural network. Again, N, C_l, H_l and W_l are respectively batch, channel, height and width axes at the corresponding layer. Batch normalization layers are indexed similarly. Since the batch size is fixed throughout the network, we drop the subscript l from N_l. For the sake of simplicity, we ignore pooling layers and assume the non-linearity activation functions to be rectified linear units (ReLU) [37], since pooling layers can be trivially added to the formulation. Similarly, other non-linearity functions can partially or fully replace ReLU throughout the network. Given these,

$$\begin{aligned} x_{l-1} &= \mathrm{ReLU}\Big(\mathrm{BN}_{l-2}(x_{l-2} * \omega_{l-2})\Big), \\ x_l &= \mathrm{ReLU}\Big(\mathrm{BN}_{l-1}(x_{l-1} * \omega_{l-1})\Big), \\ x_{l+1} &= \mathrm{ReLU}\Big(\mathrm{BN}_l(x_l * \omega_l)\Big), \end{aligned} \tag{6.13}$$

formulate three consecutive layers in the aforementioned convolution neural network where $*$ represents the convolution operation. In Section 6.1.1, given the hypothesis $x \sim \mathcal{N}(\mu, \sigma^2)$, the general form of normalization in Equation 2.6 emerges as the Fisher vector with respect to the mean of the distribution, namely $\mathcal{G}_\mu^x$. This means that the input to BN_l at the l^{th} layer in Equation 6.13, specifically $x_l * \omega_l$, is of a Gaussian distribution. Since, convolution is a linear operation, and Gaussian distribution is closed under linear combination, it further implies that $x_l \sim \mathcal{N}(\mu_l, \sigma_l^2)$. However, x_l is the output of ReLU from the $(l-1)^{th}$ layer with semi-infinite support on $[0, +\infty)$. Hence, assuming that a symmetric probability density function, like Gaussian distribution, with support on the whole Real line can model the linear combination of outputs of multiple rectified linear units, is not very well justified.

Based on this observation, we propose to parameterize the probability density function, p_θ (ref. Section 6.1.1), as a Gaussian Mixture Model (GMM). From [38], we know that any continuous distribution can be approximated with arbitrary precision using a GMM. Since our input distribution is generated by linear combination (through convolution operation) of rectified Gaussian distributions, it is not necessarily continuous and contains, at least two modes, one for the rectified values mapped to zero and one for the positive values. Therefore, we expect a mixture model to provide us with a better approximation of such distribution than a single Gaussian model.

Fisher Vector for Gaussian Mixture Model In the following, to prevent clutter, we drop subscript θ where $\theta = \{\lambda_k, \mu_k, \Sigma_k : k = 1, \dots, K\}$. Then, for $x \in \mathbb{R}^D$, without loss of generality, consider

$$p(x) = \sum_{k=1}^{K} \lambda_k p_k(x) \quad s.t. \forall_k : \lambda_k \geq 0, \sum_{k=1}^{K} \lambda_k = 1, \tag{6.14}$$

where

$$p_k(x) = \frac{1}{(2\pi)^{D/2}|\Sigma_k|^{1/2}} \exp\left\{ -\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1} (x-\mu_k) \right\}, \tag{6.15}$$

represents k^{th} Gaussian in the mixture model $p(x)$. To remove the probability simplex constraint on λ in Equation 6.14, a commonly used change of variables [144] is employed as

$$\alpha_k = \log\left(\frac{\lambda_k}{\lambda_K}\right), k = 1, \dots, K. \tag{6.16}$$

Assume $\alpha_K = 0$ to be constant, then we can re-write Equation 6.14 as

$$p(x) = \sum_{k=1}^{K} \Big(\frac{\exp(\alpha_k)}{\sum_{j=1}^{K} \exp(\alpha_j)} \Big) p_k(x). \tag{6.17}$$

Following similar derivations as in Section 6.1.1, given diagonal covariance assumption, the gradients of the log-likelihood of $p(x)$ with respect to μ_k and σ_k can be written as

$$\begin{aligned}\nabla_{\mu_k} \log p(x) &= \nu_k(x)\left(\frac{x-\mu_k}{\sigma_k^2}\right),\\ \nabla_{\sigma_k} \log p_\theta(x) &= \nu_k(x)\left(-\frac{1}{\sigma_k}+\frac{(x-\mu_k)^2}{\sigma_k^3}\right).\end{aligned} \tag{6.18}$$

where

$$\nu_k(x) = \frac{\lambda_k p_k(x)}{\sum_{j=1}^K \lambda_j p_j(x)}, \tag{6.19}$$

is the probability that x has been generated by the k^{th} Gaussian component in the mixture model. Given soft assignment distribution in Equation 6.19, Perronnin *et. al* [145] proposed a closed form approximation to the Fisher information matrix of GMM[4] as

$$F_k = \begin{bmatrix} \frac{\lambda_k}{\sigma_k^2} & 0 \\ 0 & \frac{2\lambda_k}{\sigma_k^2} \end{bmatrix}, \tag{6.20}$$

[4] For more detailed derivation of Fisher information matrix and Fisher vector in Gaussian mixture models, readers are encouraged to refer to [145] and [143].

resulting in

$$
\begin{aligned}
\mathscr{G}_k^x &= \begin{bmatrix} \mathscr{G}_{\mu_k}^x \\ \mathscr{G}_{\sigma_k}^x \end{bmatrix}, \\
&= \begin{bmatrix} \dfrac{\sigma_k}{\sqrt{\lambda_k}} & 0 \\ 0 & \dfrac{\sigma_k}{\sqrt{2\lambda_k}} \end{bmatrix} \begin{bmatrix} \nu_k(x)\left(\dfrac{x-\mu_k}{\sigma_k^2}\right) \\ \nu_k(x)\left(-\dfrac{1}{\sigma_k}+\dfrac{(x-\mu_k)^2}{\sigma_k^3}\right) \end{bmatrix}, \\
&= \frac{\nu_k(x)}{\sqrt{\lambda_k}} \begin{bmatrix} \dfrac{1}{\sigma_k}(x-\mu_k) \\ \dfrac{1}{\sqrt{2}}\left(-1+\dfrac{(x-\mu_k)^2}{\sigma_k^2}\right) \end{bmatrix}.
\end{aligned}
\tag{6.21}
$$

Comparing Equation 6.21 with Equation 6.12, we observe that Fisher vector for each component of a GMM is very similar to the one obtained from a single Gaussian distribution. However, there are two main differences:

- $\nu_k(x)$ in a soft-assignment mechanism, scales the k^{th} Fisher vector based on the posterior probability of x being generated by the corresponding Gaussian component.
- λ_k normalizes the k^{th} Fisher vector based on the contribution of the k^{th} mixture component in approximating the underlying data distribution.

We have previously shown (ref. Section 6.1.1) that the normalization formulated in batch normalization and its extensions emerges as natural kernel, *if* we assume that the underlying data distribution can be modelled by a single Gaussian distribution. We then explained that due to non-linear activation functions, such hypothesis cannot properly address the characteristics of the distribution. We employed Gaussian mixture model as an alternative and showed that natural kernel induced from GMM, follows a similar normalization mechanism, except normalization is done with respect to multiple sets of statistics obtained from different sub-populations.

Formulation Based on the aforementioned observations, we propose Mixture Normalization. Let's consider $i = (i_N, i_C, i_L)$ as a vector indexing the tensor of activations $x \in \mathbb{R}^{N \times C \times L}$ associated to a convolution layer, where the spatial domain has been flattened ($L = H \times W$). Then the *Mixture Normalizing Transform* is defined as

$$\hat{x}_i = \sum_{k=1}^{K} \frac{\nu_k(x_i)}{\sqrt{\lambda_k}} \cdot \hat{x}_i^k, \tag{6.22}$$

given

$$v_i^k = x_i - \mathbb{E}_{\mathscr{B}_i}[\hat{\nu}_k(x) \cdot x], \quad \hat{x}_i^k = \frac{v_i^k}{\sqrt{\mathbb{E}_{\mathscr{B}_i}[\hat{\nu}_k(x) \cdot (v^k)^2] + \varepsilon}}, \tag{6.23}$$

where

$$\hat{\nu}_k(x_i) = \frac{\nu_k(x_i)}{\sum_{j \in \mathscr{B}_i} \nu_k(x_j)}, \tag{6.24}$$

is the normalized contribution of x_i over $\mathscr{B}_i$ in estimating statistical measures of the k^{th} Gaussian component. Similar to batch normalization, we can also include additional parameters by slightly modifying Equation 6.23 using

$$\begin{aligned} \mathbb{E}_{\mathscr{B}_i}[\hat{\nu}_k(x) \cdot x] \quad &\rightarrow \quad \mathbb{E}_{\mathscr{B}_i}[\hat{\nu}_k(x) \cdot x] + \beta, \\ \mathbb{E}_{\mathscr{B}_i}[\hat{\nu}_k(x) \cdot (v^k)^2] \quad &\rightarrow \quad \mathbb{E}_{\mathscr{B}_i}[\hat{\nu}_k(x) \cdot (v^k)^2] \cdot \gamma, \end{aligned} \tag{6.25}$$

where β and γ respectively indicate scale and shift. Our formulation is general (*e.g.* $H, W = 1$ for fully-connected layers) and can be applied to all the variations of batch normalization, detailed in Section **??**, simply by constructing $\mathscr{B}_i$ accordingly.

During training, we fit a Gaussian mixture model to $\mathscr{B} = \{x_{1\ldots m} : m \in [1, N] \times [1, L]\}$ by Maximum Likelihood Estimation (MLE). This is a two-stage process where we use the seeding procedure of K-means++ [146] to initialize the centers of the mixture component. Then, the parameters of the mixture model, $\theta = \{\lambda_k, \mu_k, \Sigma_k : k = 1, \dots, K\}$, are estimated by Expectation-Maximization (EM) [147]. In practice, one or two EM iterations are sufficient, thanks to proper and efficient initialization by K-means++ [146]. We then normalize $x_1, x_2, \dots x_m$ with respect to the estimated parameters (Equations 6.23) and aggregate using posterior probabilities (Equations 6.22). At inference, following the batch normalization [1], we want to normalize a sample mini-batch with respect to the statistics of the training data. To do so, we can maintain last T mini-batches from the training stage. This can be simply implemented by a queue of length T. We aggregate all instances in the queue into a large pool of samples and fit a Gaussian mixture model to it. Estimated parameters are then used to normalize all the mini-batches of the test set. While being effective, the above strategy is not very desirable as it always maintains T mini-batch of samples in GPU memory. Yet, there is a very simple workaround to significantly improve this. Specifically, during training, instead of the samples, we only need to keep the estimated parameters resulting in the queue $\{\theta_0, \theta_1 \dots \theta_{T-1}\}$, where the subscript indicates the position in the queue. Note that in this case, queue always maintains $(2KC + K) \times T$ values in the GPU memory which is orders of magnitude smaller than $(mC) \times T$ of the previous strategy. The normalization will then be performed via

$$\hat{x}_i = \sum_{t=0}^{T-1} \sum_{k=1}^{K} \frac{1}{\sqrt{\tau^t \lambda_k^t}} \frac{\tau^t \lambda_k^t p_k^t(x_i)}{\sum_{q=0}^{T-1} \sum_{j=1}^{K} \tau^q \lambda_j^q p_j^q(x_i)} \Big(\frac{x_i - \mu_k^t}{\sigma_k^t}\Big), \tag{6.26}$$

where

$$\tau^t = \frac{(1-\zeta)}{(1-\zeta^T)} \zeta^{(T-t-1)} \tag{6.27}$$

is the decay factor normalized with respect to the sum of its geometric series. Here, estimated mixture models from last T mini-batch of training stage are aggregated with a scale proportional to their chronological order in the queue. By default, ζ and T are respectively set to 0.9 and 10.

Differentiability and Gradient Propagation Despite our end-to-end training, K-means++ [146] seeding procedure and EM iterations (to fit the Gaussian mixture model) are performed outside the computational graph of the neural network[5]. Hence, MN is not fully differentiable. In the following, we explain how gradient back-propagates through MN.

Let $\nu \in \mathbb{R}^{m\times K}$ be a matrix where its $(i,j)^{th}$ element represents the probability that x_i, the i^{th} row of $\mathbf{x} \in \mathbb{R}^{m\times D}$ ($\mathscr{B}$ in matrix format), has been generated by the j^{th} Gaussian component in the mixture model (Equation 6.19). The process of obtaining ν is not differentialble because of K-means++ and following EM iterations, yet we can write MN equations such that with exception of ν, it is fully differentiable.

Following Equation 6.24, we obtain the normalized contribution of each sample point x_i in estimating statistical measures of every Gaussian component via dividing each column of ν by the sum of its elements. This results in $\hat{\nu}$, the matrix of responsibilities using which we calculate $\mu = \hat{\nu}^T\mathbf{x}$ and $\sigma^2 = \hat{\nu}^T\mathbf{x}^2 - \mu^2$ where $\mu \in \mathbb{R}^{K\times D}$, $\sigma^2 \in \mathbb{R}^{K\times D}$, and λ_k is obtained by averaging the k^{th} column of ν. As we can see, all these calculations are simple differentiable operations. Hence, while gradient stops at ν, it seamlessly back-propagates through the rest of MN equations, relating parameters of the mixture component with samples and their posterior.

Effect of Non-linearity When mixture normalization is followed by ReLU [37] or similar non-linearity activation functions, rectifying should be applied to the per-component normalized activations (*i.e.* $\hat{x}_i^k$) as

$$\text{ReLU}\Big(\text{MN}(x_i)\Big) := \sum_{k=1}^{K} \frac{\nu_k(x_i)}{\sqrt{\lambda_k}} \cdot \text{ReLU}(\hat{x}_i^k), \tag{6.28}$$

where MN stands for mixture normalization. In practice, given $K \geq 3$, $\nu(x_i) = \big[\nu_1(x_i),\, \nu_2(x_i), \ldots,\, \nu_K(x_i)\big]$ is mostly very sparse, meaning that only 1 out of K elements is considerably larger than zero. As a result, we have

$$\sum_{k=1}^{K} \frac{\nu_k(x_i)}{\sqrt{\lambda_k}} \cdot \text{ReLU}(\hat{x}_i^k) \approx \text{ReLU}\Big(\sum_{k=1}^{K} \frac{\nu_k(x_i)}{\sqrt{\lambda_k}} \cdot \hat{x}_i^k\Big), \tag{6.29}$$

[5] In future, we are going to explore mixture normalization, when estimating GMM via batch and stochastic Riemannian optimization [148].

a format which is the same as how ReLU follows batch normalization. However, when Gaussian components considerably overlap, the aforementioned approximation is less accurate.

Local vs. Global Normalization Batch normalization is an Affine transform on the *whole* probability density function of the underlying distribution. In other words, *all* the samples from the distribution are normalized using the *same* mean and standard deviation, estimated from the entire population. In contrast, in proposed mixture normalization, we transform each sample using mean and standard deviation of the mixture component to which the sample belongs to. Therefore, one can see mixture normalization as a soft piecewise normalizing transform. Figure 6.1 illustrates the aforementioned differences. Note how in case of batch normalization, $p(x)$, modeled by a Gaussian density function, fails to properly approximate the underlying data distribution. This problem is more severe when the distribution is skewed. Mixture normalization instead, is capable of handling these challenges as it disentangles the modes of variation. In Figure 6.1, we show mixture normalization with K set to 2 and 3. Obviously, when the number of components is set to 1, mixture normalization reduces to the general form of normalization. We observe that even with two components, mixture normalization provides a better approximation to the underlying data distribution. However, in some cases (*e.g.* columns correspond to the 4^{th} and 28^{th} channels), we can benefit from increasing the number of mixture components. Mixture normalized activations, in comparison to the batch normalized ones, are considerably closer to the normal distribution, and illustrate less skewed probability densities.

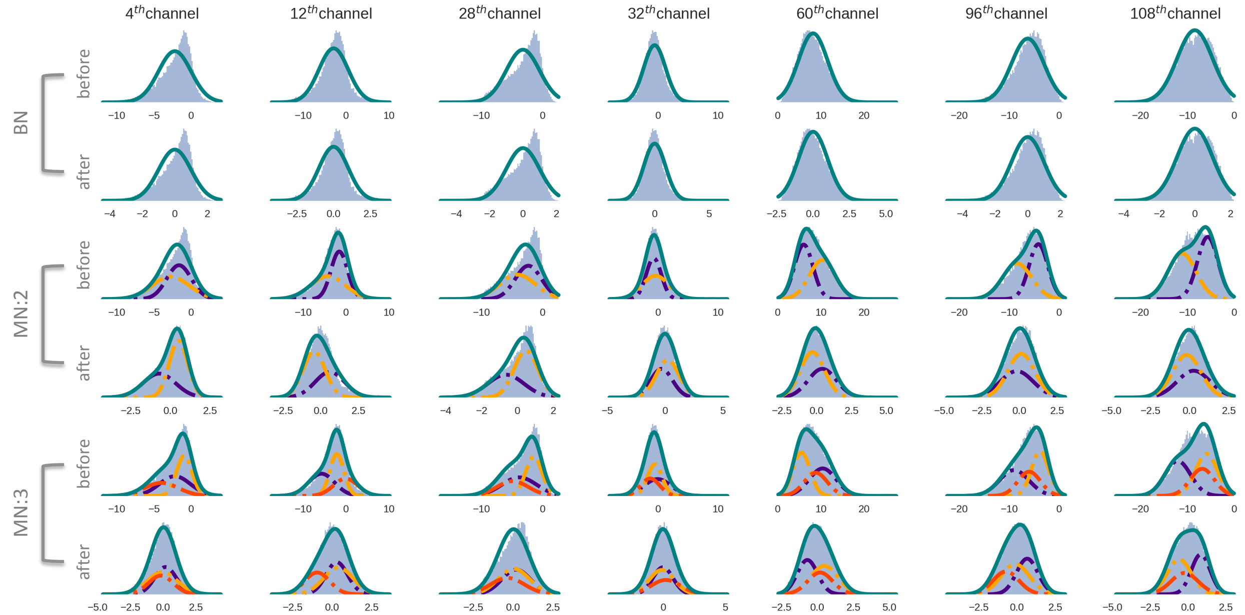


Figure 6.1: Visualizing Mixture Normalization: Given a random mini-batch in the midway of training on CIFAR-100, we illustrate the underlying distribution of activations (output of convolution) associated with a random subset of 128 channels in the layer "conv2" of CIFAR CNN architecture (detailed in Table 6.1). Solid teal curve indicates the probability density function. Dashed curves represent different mixture components shown in various colors. Note that similar colors across multiple subfigures, simply index mixture components and do not indicate any association. We observe that mixture normalization, shown by MN:2 and MN:3 (2 and 3 respectively represent the number of components in the mixture of Gaussians), provides better approximation, $p(x)$, illustrated by solid teal curve, to the underlying distribution. Also, mixture normalized activations, in comparison to the batch normalized ones, are considerably closer to the normal distribution and illustrate less skewed probability densities.

6.2 Experiments

To evaluate the effectiveness of our proposed mixture normalization, we conduct extensive set of experiments on CIFAR-10 and CIFAR-100 datasets [149]. We compare mixture normalization against its batch normalization counterpart in a variety of settings given four backbone choices, namely a 5-layers deep fully convolutional neural network, the Inception-V3 [26], along with the 40-layers and 100-layers deep DenseNet [7] architectures. According to the literature [35][36][39], when mini-batch size is sufficiently large (*e.g* 32), standard batch normalization [1] outperforms its variants in image classification. Although, this is not valid in recurrent networks. Therefore, here, we only compare against standard batch normalization [1].

It is important to emphasize, that we do not aim at achieving state-of-the-art results as it requires employing computationally expensive architectures, and involves careful tuning of many hyper-parameters and heuristics. Instead, we are interested in understanding the behavior of mixture normalization. We focus on demonstrating that solely replacing *a few* batch normalization with our proposed mixture normalization, can dramatically increase the convergence rate, and in majority of cases even yields a better final test accuracy.

In summary, in this section:

- We compare BN and MN, in small and large learning rate regimes, using both shallow and very deep CNN architectures.
- We show the effect of applying MN to different layers, while varying the number of mixture components and EM iterations.
- We quantify the acceleration obtained using MN, by measuring required number of gradient updates for mixture normalized model, in order to reach the best test accuracy of its batch normalized counterpart.
- We demonstrate, that our findings are consistent with respect to the choice of the optimization technique (adaptive v.s non-adaptive) and learning rate decay policies.
- We finish this section by demonstrating the application of mixture normalization in Generative adversarial networks (GANs) [150].

6.2.1 Datasets

For our experiments, we use popular CIFAR [149] datasets. Images are 32×32 and labeled with 10 and 100 classes, respectively for CIFAR-10 and CIFAR-100. We follow the standard data split where 50K images are used for training and 10K for testing. As for preprocessing, we normalize images with respect to the mean and standard deviation of the dataset. Also, similar to the previous works [32][7][151][152][24] on CIFAR, we adopt horizontal flipping and random cropping for data augmentation.

6.2.2 CIFAR CNN

We begin with a small 5-layers deep convolutional neural network architecture detailed in Table 6.1, where we later replace batch normalization in certain layers with mixture normalization. Following [1], we experiment using two different learning rates, one 5 times larger than the other. Additionally, we do the same with the weight decay to very well cover various training regimes. We use RMSprop [6] with 0.9 momentum and exponentially reduce the initial learning rate every two epochs with the decay rate of 0.93. The size of mini-batch is set to 256 and we train all the models for 100 epochs. To facilitate comparison between different training settings, we index experiments accordingly, where BN and MN, respectively, indicate usage of batch and mixture normalization.

In the first set of experiments, for different mixture normalization variations, shown in Table 6.2 by MN-1 to MN-4, we replace the batch normalization in the "conv3" layer (ref. Table 6.1) with mixture normalization while keeping the remaining layers intact. From Table 6.2 and Figure 6.2, we observe that irrespective of the weight decay and learning rates, not only MN models converge much faster than their corresponding BN counterparts, but they achieve a better test accuracy after

100 epochs.

Table 6.1: CIFAR CNN architecture

layer	type	size	kernel	(stride, pad)
input	input	$3 \times 32 \times 32$	–	–
conv1	conv+bn+relu	$64 \times 32 \times 32$	5×5	(1, 2)
pool1	max pool	$64 \times 16 \times 16$	3×3	(2, 0)
conv2	conv+bn+relu	$128 \times 16 \times 16$	5×5	(1, 2)
pool2	max pool	$128 \times 8 \times 8$	3×3	(2, 0)
conv3	conv+bn+relu	$128 \times 8 \times 8$	5×5	(1, 2)
pool3	max pool	$128 \times 4 \times 4$	3×3	(2, 0)
conv4	conv+bn+relu	$256 \times 4 \times 4$	3×3	(1, 1)
pool4	avg pool	$256 \times 1 \times 1$	4×4	(1, 0)
linear	linear	10 or 100	–	–

Large learning rate: When we increase the learning rate from 0.01 to 0.05, the convergence gap is even larger (ref. Table 6.2 and Figure 6.2), demonstrating the capability of mixture normalization to better utilize larger learning rates for training. Note, that for the large learning rate regime to eventually match the final learning rate of small learning rate regime, either the exponential decay rate should be reduced (*e.g* 0.91 instead of 0.93) or the number of times it is applied must be increased (*e.g* every epoch instead of every 2 epochs). However, here we keep the same learning rate decay policy to probe the sole effect of increasing the learning rate.

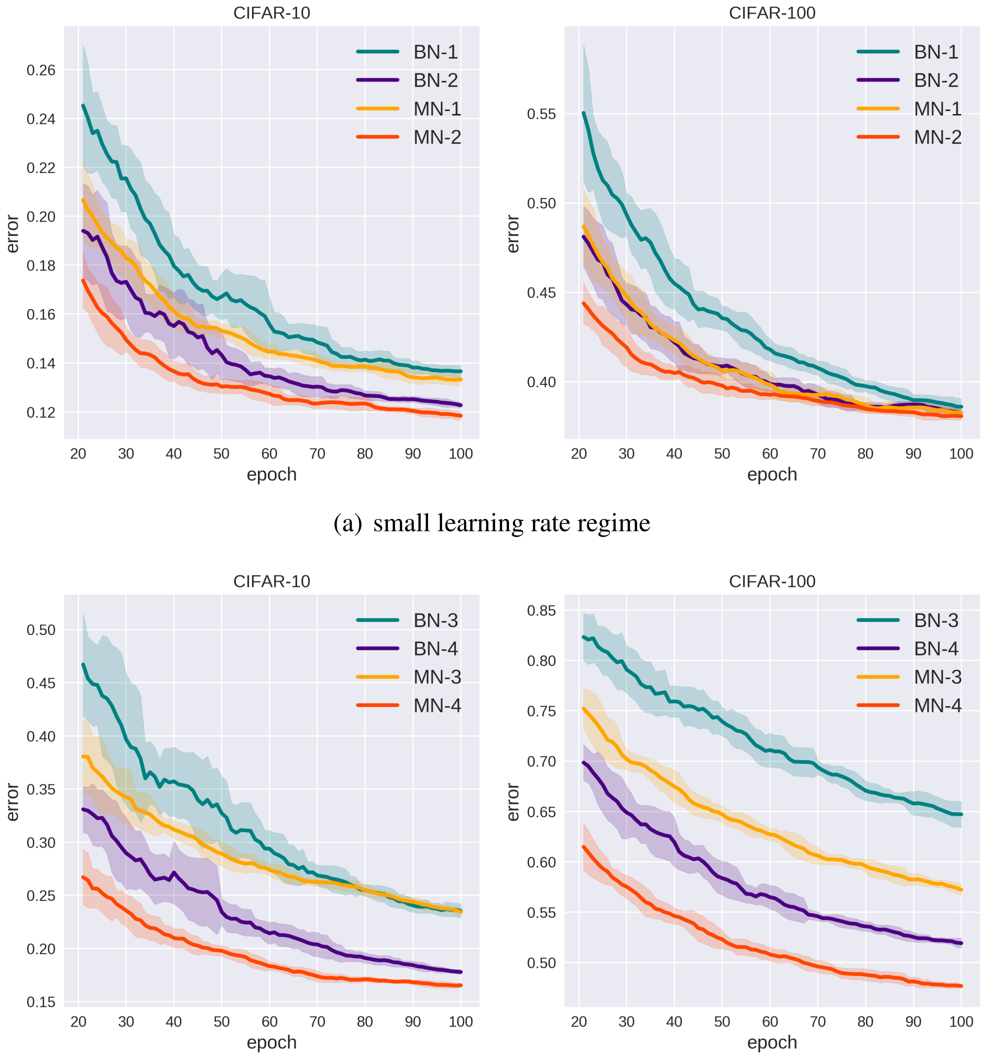
Stability: Another observation is with regards to the uncertainty of the model predictions. Figure 6.2 shows one standard deviation (shaded area) computed within a window of 10 epochs for all the test error curves. We see mixture-normalized models illustrate a considerably more stable test error, meaning from one epoch to another, it is less likely that the performance fluctuates abruptly. This alongside faster convergence [153] is reminiscent of relatively flat minima in the optimization landscape [154], where the classification margin of the model, as a whole, is robust with respect to small changes (gradient updates) to the model parameters. Note that in all these experiments, we have solely replaced one batch normalization layer with mixture normalization.

Table 6.2: Experiments on CIFAR-10 and CIFAR-100 using CIFAR CNN architecture (ref. Table 6.1). We observe that irrespective of the weight decay and learning rate, not only MN models converge faster but also achieve better final test accuracy, compared to their corresponding BN counterparts. When mixture normalization is applied to multiple layers (*i.e* MN-8), we use the same K and EM iter. values for all the corresponding layers.

CIFAR-10						
	mixture norm. setting	training setting	maximum test accuracy(%) after			
model	(layer, K, EM iter.)	(lr, weight decay)	25 epochs	50 epochs	75 epochs	100 epochs
BN-1	–	(0.01, 1e-4)	79.37	84.34	86.24	86.74
BN-2	–	(0.01, 2e-5)	84.11	86.92	87.65	87.95
BN-3	–	(0.05, 1e-4)	68.77	71.68	75.25	77.35
BN-4	–	(0.05, 2e-5)	73.84	77.84	81.43	82.58
MN-1	(conv3, 3, 2)	(0.01, 1e-4)	81.91	85.10	86.51	87.08
MN-2	(conv3, 3, 2)	(0.01, 2e-5)	85.56	87.28	88.05	88.47
MN-3	(conv3, 3, 2)	(0.05, 1e-4)	67.84	73.09	74.80	77.33
MN-4	(conv3, 3, 2)	(0.05, 2e-5)	76.70	80.82	83.31	83.87

CIFAR-100						
	mixture norm. setting	training setting	maximum test accuracy(%) after			
model	(layer, K, EM iter.)	(lr, weight decay)	25 epochs	50 epochs	75 epochs	100 epochs
BN-1	–	(0.01, 1e-4)	52.54	57.83	60.37	62.11
BN-2	–	(0.01, 2e-5)	56.09	60.05	61.46	62.14
BN-3	–	(0.05, 1e-4)	23.13	28.12	33.21	36.80
BN-4	–	(0.05, 2e-5)	36.98	43.89	46.95	49.08
MN-1	(conv3, 3, 2)	(0.01, 1e-4)	55.30	60.47	61.46	62.20
MN-2	(conv3, 3, 2)	(0.01, 2e-5)	59.04	60.86	61.77	62.29
MN-3	(conv3, 3, 2)	(0.05, 1e-4)	29.80	36.30	40.66	43.77
MN-4	(conv3, 3, 2)	(0.05, 2e-5)	42.31	48.96	51.98	52.62
MN-5	(conv3, 3, 4)	(0.01, 2e-5)	59.05	61.94	62.56	62.76
MN-6	(conv3, 3, 8)	(0.01, 2e-5)	58.98	61.83	62.67	63.12
MN-7	(conv2, 3, 2)	(0.01, 2e-5)	57.63	61.23	62.15	62.63
MN-8	((conv2,conv3), 3, 2)	(0.01, 2e-5)	59.79	61.67	62.50	62.96

Effect of parameters: Solving for the parameters of GMM begins with an initialization using K-means clustering. To obtain the best initial seeds, we use $2 + \log(K)$ trials and then perform standard K-means clustering for a certain number of iterations. The result is then used to initiate the GMM parameters, which is later followed by a fixed number of EM updates.



(a) small learning rate regime

(b) large learning rate regime

Figure 6.2: Test error curves when CIFAR CNN architecture (ref. Table 6.1) is trained under different learning rate and weight decay settings. We observe that on CIFAR-10 and CIFAR-100, MN performs consistently in both small and large learning rate regimes.

In our experiments, we use the same number of iterations at two phases and report their summation as “EM iter.” in Table 6.2 (and later in Table 6.4). From Figure 6.3 (left) and Table 6.2, we observe that more EM iterations, provides faster convergence and better final test accuracy.

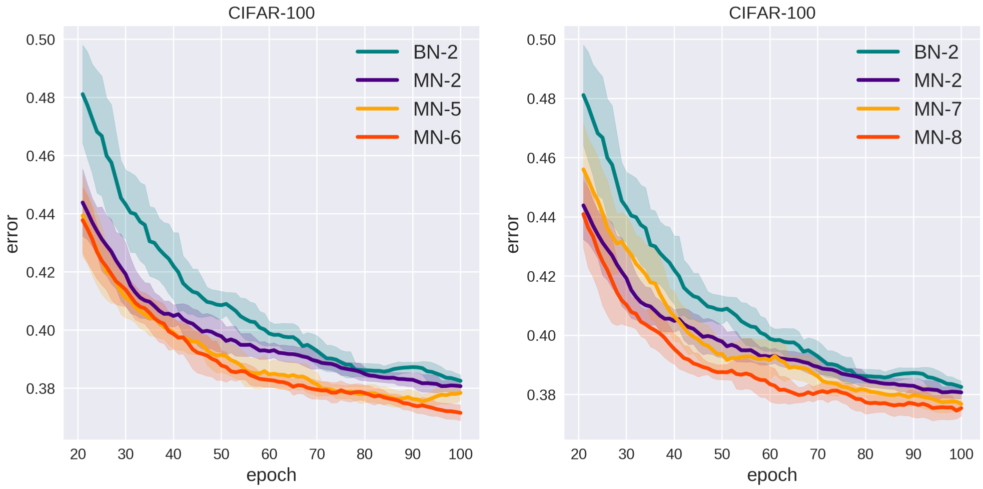


Figure 6.3: Left: effect of the number of EM iterations on test error. Right: effect of utilizing MN at different layers, on test error. We show that more EM iterations and utilizing MN at multiple layers, increase the convergence rate of mixture normalized models.

This is aligned with our expectation as more EM iterations translates to better approximation of the underlying data distribution. However, the downside is that more EM iterations results in increasing the computation time. So far, we have only modified one layer in CIFAR CNN architecture. In Figure 6.3 (right), we illustrate that similar behavior can be observed when mixture normalization is employed at earlier layers like “conv2”. Furthermore, model enjoys even faster convergence rate in addition to better final test accuracy (ref. Table 6.2), when more layers (“conv2” and “conv3”) are equipped with mixture normalization.

Quantifying Acceleration: We illustrate in Figure 6.2, that indeed using mixture normalization accelerates the convergence and results in lower final test error. Table 6.2 also indicates the same when we report the maximum test accuracy after 25%, 50%, 75% and 100% of total number of epochs. To provide a precise comparison, we follow Ioffe and Szegedy [1] and report the number of steps (gradient descent updates) for mixture normalization variants in order to reach the maximum test accuracy achieved by their batch-normalized counterparts.

Table 6.3: For batch normalization and the mixture-normalized variants using CIFAR CNN architecture (ref. Table 6.1), the number of training steps required to reach the maximum accuracy of batch-normalized model alongside with the maximum accuracy achieved by each variant.

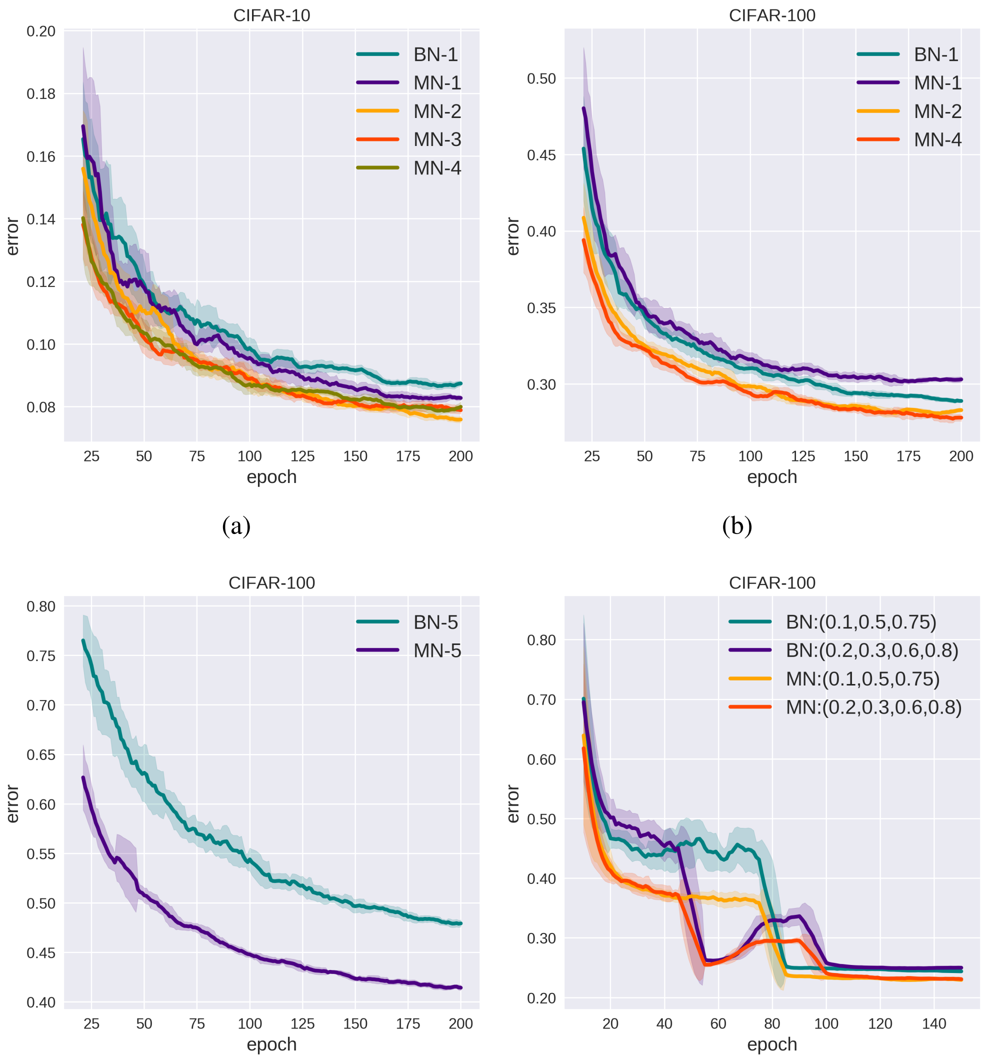
CIFAR-10		
model	steps to 87.95%	max. acc.(%)
BN-2	1.95×10^4	87.95
MN-2	1.34×10^4	88.47
CIFAR-100		
model	steps to 62.14%	max. acc.(%)
BN-2	1.81×10^4	62.14
MN-2	1.77×10^4	62.29
MN-5	1.07×10^4	62.76
MN-6	1.09×10^4	63.12
MN-7	1.34×10^4	62.63
MN-8	1.03×10^4	62.96

Table 6.3 shows that on CIFAR-10, mixture normalization reduces the number of training steps in order to reach 87.95% test accuracy by ∼**31**%. Similarly, on CIFAR-100, the best performing variant of mixture normalization, MN-6, reduces the number of training steps in order to reach 62.14% test accuracy by ∼**40**%, meanwhile its very own maximum test accuracy outperforms BN-2 by ∼1%. This further affirms that, our proposed mixture normalization is not only effectively accelerating the training procedure, but also reaches better local minima.

6.2.3 Inception-V3

So far, we have shown that mixture normalization can improve the convergence rate of batch-normalized models. However, our experiments were conducted on a shallow 5-layers deep architecture. Hence, it is reasonable to question whether the same behavior can be observed in very deep and more modern architectures? To address this, we choose Inception-V3 [26]. Its architecture is 48 layers deep and uses global average pooling instead of fully-connected layers, which allows operating on arbitrary input image sizes. Inception-V3 [26] has a total output stride of 32. However, to maintain low computation cost and memory utilization, the size of activation maps quickly reduces by a factor of 8 in only first seven layers. This is done by one convolution and two max pooling layers that operate with the stride of 2. The network is followed by three blocks of Inception separated by two grid reduction modules. Each Inception block consists of multiple Inception layers that are sequentially stacked. Specifically, first, second and third Inception blocks are respectively comprised of 3, 4 and 2 Inception layers. Spatial resolution of the activations remains intact within the Inception blocks, while grid reduction modules halve the activation size and increase the number of channels.

To make Inception-V3 [26] architecture effectively applicable to images in CIFAR-10 and CIFAR-100, that are only 32×32, we need to slightly modify the architecture. Specifically, we change the stride of the first convolution layer from 2 to 1 and remove the first max pooling layer. This way, the output stride of Inception-V3 architecture reduces to 8. That is, activations maintain sufficient resolution throughout the network's depth, with the final activation (before the global average pooling) be of size 3×3. From now on, when we refer to the Inception-V3, we mean this modified version. To train our models, we use RMSprop [6] with 0.9 as momentum and exponentially reduce the initial learning rate every four epochs with the decay rate of 0.93. The size of mini-batch is set to 128, weight decay to 0.0005 and we train all the models for 200 epochs. In our preliminary experiments, we observed that learning rate of 0.001 gives the best final test accuracy for batch normalized models. Therefore, we use it for all the experiments except in one case which aims at analyzing large learning rate regime.



(a)

(b)

(c)

(d)

Figure 6.4: Test error curves when Inception-V3 architecture is trained under different settings. Figures 6.4(a) and 6.4(b) show the small learning rate regime, respectively, on CIFAR-10 and CIFAR-100. Figure 6.4(c) shows the large learning rate regime on CIFAR-100. Figure 6.4(d) illustrates test error curves of CIFAR-100 when Inception-V3 architecture is trained using Nesterov's accelerated gradient [5] (all other experiments use RMSprop [6]), with two different learning rate drop policies. Mixture normalization modules have been employed in "inc2/1" and "inc3/0" layers. We observe that across a variety of choices such as the number of mixture components, number of EM iterations, learning rate regime and drop policy, optimization technique, and the layer where MN is applied, mixture normalized models consistently accelerate their batch normalized counterparts and achieve better final test accuracy.

Table 6.4: Experiments on CIFAR-10 and CIFAR-100 using Inception-V3 architecture. Notation: "red1" ("red2") refers to the first (second) grid reduction modules. Similarly "inc2/0" ("inc3/0") refers to the first inception layer in second (third) inception block of the architecture. In MN-$*$ settings, we only replace the last batch normalization in each branch of the corresponding Inception layer with our mixture normalization. When mixture normalization is applied to multiple layers, we use the same K and EM iter. values for all the corresponding layers.

			CIFAR-10			
	mixture norm. setting	training setting	maximum test accuracy(%) after			
model	(layer, K, EM iter.)	(lr, weight decay)	50 epochs	100 epochs	150 epochs	200 epochs
BN-1	–	(0.001, 5e-4)	89.18	90.62	91.08	91.50
MN-1	((red1,red2), 3, 8)	(0.001, 5e-4)	89.47	91.03	91.70	91.91
MN-2	((red1,red2,inc3/0), 3, 4)	(0.001, 5e-4)	89.73	91.69	92.09	92.55
MN-3	((inc2/0,inc3/0), 3, 2)	(0.001, 5e-4)	90.49	91.52	92.17	92.30
MN-4	((inc2/0,inc3/0), 5, 2)	(0.001, 5e-4)	90.26	91.77	92.00	92.25
			CIFAR-100			
	mixture norm. setting	training setting	maximum test accuracy(%) after			
model	(layer, K, EM iter.)	(lr, weight decay)	50 epochs	100 epochs	150 epochs	200 epochs
BN-1	–	(0.001, 5e-4)	66.88	69.44	71.03	71.30
BN-5	–	(0.005, 5e-4)	40.68	47.47	51.13	52.63
MN-1	((red1,red2), 3, 8)	(0.001, 5e-4)	66.6	68.86	70.01	70.43
MN-2	((red1,red2,inc3/0), 3, 4)	(0.001, 5e-4)	68.26	70.78	71.78	72.23
MN-4	((inc2/0,inc3/0), 5, 2)	(0.001, 5e-4)	68.4	70.99	71.98	72.74
MN-5	((inc2/0,inc3/0), 5, 2)	(0.005, 5e-4)	50.41	55.69	57.88	59.10

Table 6.4 and Figure 6.4 compare mixture normalization with its standard batch normalization counterparts. We observe that, with the exception of MN-1 on CIFAR-100, all the MN variants are not only successfully accelerating BN variants, but also achieve superior final test accuracy. As expected, and similar to the shallow network case, there is a benefit in using mixture normalization at multiple layers in different depth. Table 6.4 also recommends using mixture normalization in later layers, closer to the classifier, to reach a better convergence rate and final test accuracy.

Table 6.5: For batch normalization and the mixture-normalized variants using Inception-V3, the number of training steps required to reach the maximum accuracy of batch-normalized model, and the maximum accuracy achieved by each variant.

CIFAR-10		
model	steps to 91.50%	max. acc.(%)
BN-1	7.34×10^4	91.50
MN-1	4.29×10^4	91.91
MN-2	3.86×10^4	92.55
MN-3	3.82×10^4	92.30
MN-4	3.58×10^4	92.25
CIFAR-100		
model	steps to 71.30%	max. acc.(%)
BN-1	7.34×10^4	71.30
MN-1	–	70.43
MN-2	4.57×10^4	72.23
MN-4	3.93×10^4	72.74

Table 6.6: Training Inception-V3 using Nesterov's accelerated gradient [5] on CIFAR-100, the number of training steps required to reach the maximum accuracy of batch-normalized model along with the maximum accuracy achieved by each model.

lr decay policy: (0.1,0.5,0.75)		
model	steps to 75.75%	max. acc.(%)
BN	2.78×10^4	75.75
MN	1.52×10^4	77.30
lr decay policy: (0.2,0.3,0.6,0.8)		
model	steps to 75.20%	max. acc.(%)
BN	2.41×10^4	75.20
MN	1.79×10^4	77.32

Effect of parameters: We can observe from MN-3 versus MN-4 on CIFAR-10, that mixture normalization's performance is not sensitive to the number of mixture components assuming K is large enough. We have discussed above that the number of EM iterations directly affect the computation time. However, Table 6.4 and Figure 6.4, demonstrate that by employing mixture normalization even with 2 EM updates, training a modern 48-layers deep architecture such as Inception-V3 will enjoy decent acceleration.

Large learning rate: To analyze the behavior of mixture normalization in large learning rate regime, we experiment training our models with the learning rate of 0.005 (5×0.001). Figure 6.4(c) illustrates that MN-5 handles large learning rate better than BN-5, its batch normalized counterpart. Note that, to probe the sole effect of increasing the learning rate, here we keep the same learning rate decay policy (decay rate of 0.93 every four epochs) as the small learning rate regime.

Quantifying Acceleration: Similar to the case of CIFAR CNN architecture, we report the number of steps (gradient descent updates) mixture normalization variants require in order to reach the maximum test accuracy obtained by their batch-normalized counterparts. Table 6.5 shows that the best performing variants of mixture normalization, MN-2 on CIFAR-10 and MN-4 on CIFAR-100, reduce the number of training steps towards 91.50% and 71.30% test accuracy, respectively on CIFAR-10 and CIFAR-100, by $\sim$**47%**. Similar to the case of shallow CIFAR CNN architecture, here, mixture normalization, also improves the final test accuracy by $\sim$1-1.5%.

Choice of optimization technique: We evaluate whether the performance of mixture normalization is consistent using different choices of optimization technique. That is, instead of RMSprop [6], we use Nesterov's accelerated gradient [5], with 0.9 as momentum and decay the learning rate in two different fashions. First, following the policy adopted in ResNet [32] and DenseNet [7], we reduce the learning rate twice by a factor of 10 at 50% and 75% of the total number of epochs.

Second, following Wide Residual networks [151], we reduce the learning rate three times with a factor of 5 at 30%, 60% and 80% of the total number of epochs. For MN models, mixture normalization is employed at “inc2/1” and “inc3/0” layers. Mini-batch size and the initial learning rate are respectively set to 256 and 0.14 and we train all the models for 150 epochs.

Figure 6.4(d) shows the test error curves on CIFAR-100. Similar to the previous experiments, in Table 6.6, we compare the required number of gradient updates in order to reach the maximum test accuracy of batch normalized model. In both learning rate decay scenarios, mixture normalization is not only able to considerably accelerate training procedure, but also achieves $\sim$2% better final test accuracy. Finally, in comparison with the state-of-the-art architectures of comparable depth, our MN model, with a test error of **22.68**% on CIFAR-100, performs on par with Wide Residual networks [151] while outperforming DenseNet [7] (ref. Table 2 in [7]).

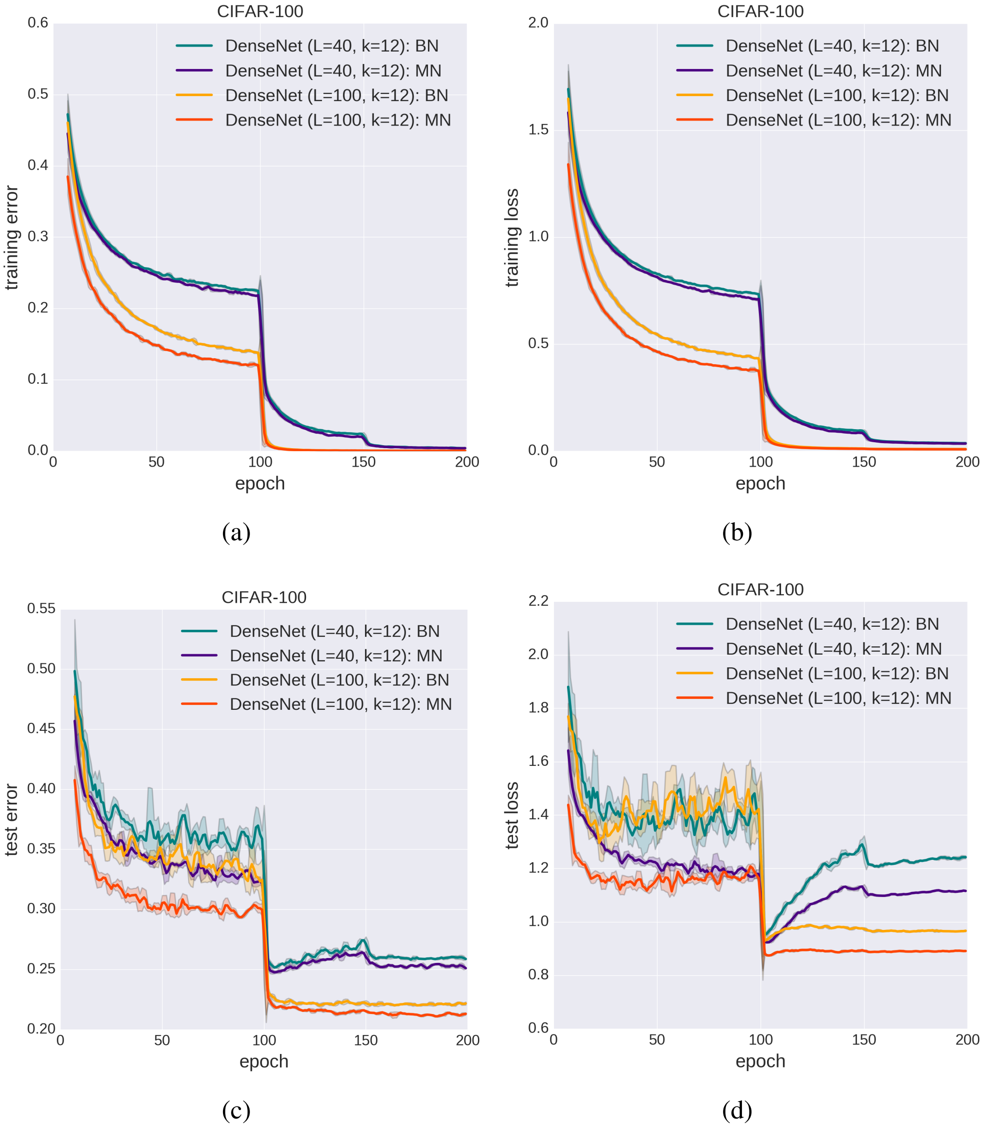
6.2.4 DenseNet

We conclude our experimental results by evaluating the effectiveness of mixture normalization in DenseNet [7] architectures. We use two basic architectures with 40 and 100 layers. There are three dense blocks and two transition layers. Both architectures are using a growth rate of 12. All models are trained for 200 epochs on CIFAR-100, with the batch size of 64 using Nesterov’s accelerated gradient [5]. Learning rate is initialized at 0.1 and is divided by 10 at 50% and 75% of the total number of training epochs. We set the weight decay and momentum values to 10^{-4} and 0.9, respectively. Figure 6.5 illustrates the classification error and cross entropy loss of training and test. For mixture normalized models, denoted by MN, we solely replace the batch normalization layers of two transition layers and the last (after the third dense block), with mixture normalization. The number of components and EM iterations for MN variants are set to 5 and 2, respectively. From Figure 6.5, we observe that mixture normalization not only facilitates training by accelerating the optimization, but also consistently provides better generalization on both architecture settings. In fact, the benefit of mixture normalization is more clear on the deeper setting of the DenseNet architecture. Using DenseNet {L=40, k=12} on CIFAR-100, the best test error achieved by the BN variant is **25.10**% versus **24.63**% of its MN counterpart. When we switch to deeper DenseNet {L=100, k=12}, architecture, this number reduces to **21.93**% for BN-based model while MN variant reaches the best test error of **20.97**%.

6.2.5 Mixture Normalization in GANs

Generative adversarial networks (GANs) [150] have recently shown amazing progress in generating new realistic images [8][155][156][157]. The training process is a minimax game between generator and discriminator. Discriminator learns to separate real images from the fake ones, created by the generator. Meanwhile, generator attempts to impede discriminator's job by progressively generating more realistic images. Hence, at convergence, generator, theoretically, must be able to generate images, whose distribution matches the distribution of the real images. However, in practice, "mode collapse"[158][159][160][161][162][163] problem prevents generator from learning a diverse set of modes with high probability, as is in the distribution of real images. Therefore, since our proposed mixture normalization, normalizes internal activations, independently over multiple disentangled modes of variation, we hypothesize that employing it in generator, should improve the training procedure of GANs.

To evaluate our hypothesis, we consider popular Deep Convolutional GAN (DCGAN) [8] architecture. Its generator consist of one linear and four deconvolution layers. The first three deconvolution layers are separated from each other by batch normalization [1] followed by ReLU[37] activation function. We replace the batch normalization layers associated with the first two deconvolution layers with mixture normalization (K=3, EM iter=2).



(a)

(b)

(c)

(d)

Figure 6.5: DenseNet [7] experiments on CIFAR-100. Figures 6.5(a) and 6.5(b) respectively illustrate the training error and cross entropy loss. Figures 6.5(c) and 6.5(d) respectively illustrate the test error and cross entropy loss. We observe from 6.5(a) and 6.5(b) that mixture normalization facilitates the training process by accelerating the optimization. Meanwhile it provides better generalization (ref. 6.5(c), and 6.5(d)) by continuously maintaining a large gap with respect to its batch normalized counterpart. We show 1 standard deviation (shaded area) computed within a window of 3 epochs for all the curves.

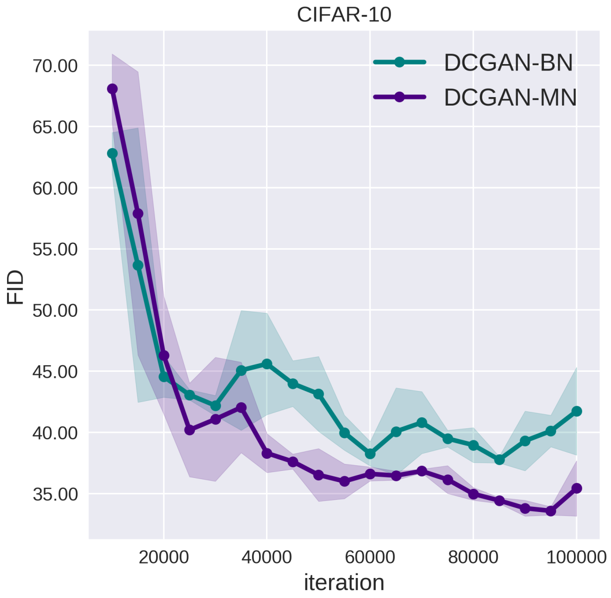
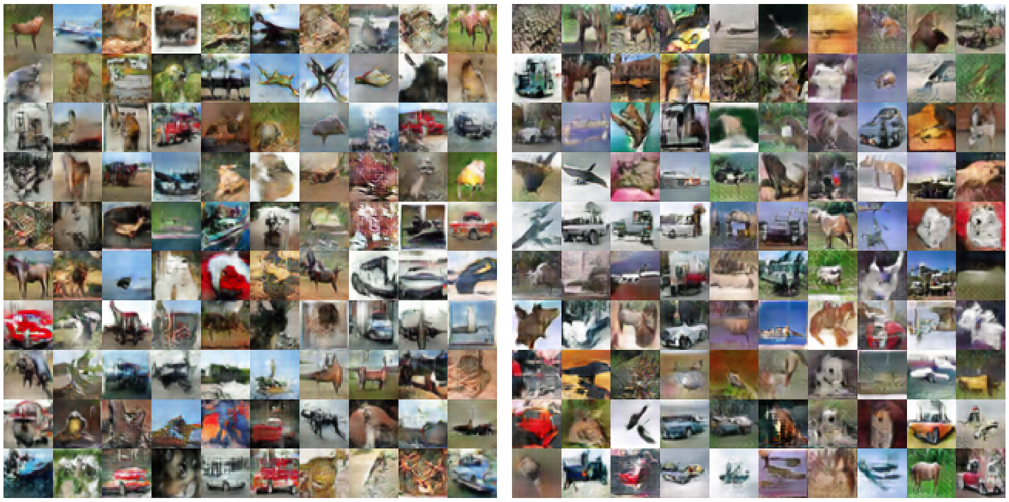


Figure 6.6: Mixture normalization in deep convolutional GAN (DCGAN)[8]. We observe that employing our proposed mixture normalization in the generator of DCGAN (DCGAN-MN) facilitates the training process. In comparison with the standard DCGAN which uses batch normalization (DCGAN-BN), DCGAN-MN not only converges faster (a reduction of ∼58%) but also achieves better (lower) FID (33.35 versus 37.56). For better visualization, we show one standard deviation (shaded area) computed within a window of 30K iterations (3 adjacent FID evaluation points).

We train all the models on CIFAR-10 for 100K updates (iterations) using Adam [164] with $\alpha = 0.0002$, $\beta_1 = 0$ and $\beta_2 = 0.9$ for both generator and discriminator. The quality of GANs are measured using "Fréchet Inception Distance"(FID) [165], evaluated every 10K updates for computational efficiency. Figure 6.6 demonstrates that mixture normalized DCGAN (DCGAN-MN) not only converges faster than its batch normalized counterpart but also achieves better (lower) FID. While DCGAN-BN reaches the lowest FID of 37.56 (very close to 37.7 reported in [165]) after 60K steps (gradient updates). It only takes 25K steps, a reduction of ∼58%, for DCGAN-MN to reach 37.56. Furthermore, the lowest FID obtained by DCGAN-MN is 33.35, a significant improvement over the batch normalized model. Figure 6.7 illustrates samples of generated images by batch and mixture normalized DCGANs at their lowest evaluated FID.



(a) DCGAN-BN (b) DCGAN-MN

Figure 6.7: Samples of generated images by batch and mixture normalized DCGAN models, at their best (lowest) evaluated FID, are respectively illustrated in Figure 6.7(a) and 6.7(b). DCGAN-BN and DCGAN-MN, respectively achieve FID of 37.56 and 33.35.

6.3 Computational Complexity and Detailed Analysis

In this section, we provide in-depth analysis on the behavior of the mixture normalization. Specifically, we provide computational complexity analysis of mixture normalization, both in theory and practice. We discuss how representation of mixture components evolves and whether the rationale to utilize mixture model remains valid as we pass early stages of the training process.

6.3.1 Computational Complexity Analysis

The computational overhead of mixture normalization, in comparison to the batch normalization [1], is in estimating the parameters of the Gaussian mixture model. This is a two-stage process where, we use a seeding procedure to initialize the centers of the mixture components, followed by estimating the parameters of the mixture model through iterations of Expectation-Maximization

(EM).

In our implementation[6], we use K-means++ [146] as the seeding procedure. Without any assumption on the data, it is in expectation $O(\log k)$-competitive with the optimal K-means clustering solution [146], and has a complexity of $\Theta(nkd)$ where, n is the number of data points, k is the number of cluster centers and d is the dimensionality of the data [168]. However, there is a rich literature on algorithms that speed up K-means++ [146] seeding procedure. For example, Bachem *et. al* [168] have proposed a MCMC-based sampler to approximate the full seeding step of K-means++ with complexity of $\Theta(mk^2d)$. Under some light assumptions, the authors prove that their proposed solution is in expectation $O(\log k)$-competitive with the optimal solution, if we have the chain length $m \in \Theta(k \log^2 n \log k)$. In this case, the total computational complexity will be $O(k^3 dn \log k)$, which is sublinear in the number of data points. Authors in [169] show that an assumption-free version of [168] with lower complexity is also achievable. Finally, to ameliorate the inherent sequential passes over data, Scalable K-means++ [170] with nice theoretical support [171] can be used. These all indicate that the initial seeding procedure used in mixture normalization can be implemented with low computational complexity while maintaining provably good seeding. It is easy to show that assuming diagonal covariance, each EM iteration to fit the GMM parameters is of complexity $O(nkd)$ [172], where n can significantly be reduced using coresets [173]. In practice, we found that one does not need to use all the data points to estimate the parameters of the mixture model. Instead a simple random sampling that maintains at least 25% of the data points is sufficiently accurate. Hence, we confirm that estimating the parameters of the Gaussian mixture model can be done efficiently in scale.

In order to see how the above complexity analysis translates in practice, we compared the computation cost of mixture normalization against native implementation of batch normalization [1].

[6] Mixture Normalization is implemented in Chainer [166], and thanks to CUPY [167], the entire computation including K-means++ and EM iterations are performed in GPU.

Table 6.7: Computation cost comparison of mixture normalization against natively implemented (no CUDA-kernel) batch normalization [1]. Experiments are conducted on a Titan X (Pascal) GPU.

model	K	batch size	dim.	iteration/sec.
BN	1	64	196	16.40
MN	2	64	196	14.46
MN	3	64	196	14.42
MN	4	64	196	13.91
MN	5	64	196	13.69
BN	1	64	448	9.26
MN	2	64	448	8.83
MN	3	64	448	8.51
MN	4	64	448	8.52
MN	5	64	448	8.41
BN	1	128	448	7.35
MN	2	128	448	6.59
MN	3	128	448	6.51
MN	4	128	448	6.41
MN	5	128	448	6.38

Experiments are conducted on CIFAR-100 using Densenet [7] architecture where, MN replaces the last (after the third dense block) BN layer. We tried Densenet (3 blocks and growth rate of 12) with 20 and 40 layers which result in the number of input channels to batch/mixture normalization to be respectively 196 and 448. This allows us to evaluate the effect of dimensionality of data points. We also used batch sizes of 64 and 128 in order to study the effect of the number of data points. Finally, we varied the number of mixture components (K) from 2 to 5 to analyze its effect on the computation cost. Note that the iteration/sec. counts for the computation of the entire network not just the batch/mixture normalization layer. Table 6.7 indicates that in practice, the computation cost of MN scales very well with respect to the number of datapoints, dimensionality of data and the number of mixture components.

6.3.2 Evolution of Mixture Components

To better understand how mixture normalized models evolve, we visualize in Figure 6.8, the mixture components associated to "inc2/0/2/4"[7] layer as training MN-4 (ref. Table 6.4) on CIFAR-100 progresses. We show a random subset of 192 channels at 20%, 40%, 60%, 80% and 100% of total training iterations.

There are two main observations here. First, in early stages of training, multiples modes captured by different mixture components are of relatively similar weights (λ_k in Equation 6.14). That is aligned with our argument that due to non-linearities, underlying distributions are comprised of multiple modes of variation. However, as training procedure goes on, mixture components evolve as some get closer, while others are pushed away from each other creating more distinct components. Second, notice how the horizontal axis, associated with the activation values, reduces in range. This alongside with distribution of λ_ks morphing from relatively uniform into one with a dominant bin (component in olive), demonstrates that mixture normalization, as intended, *tries* to transform the underlying distribution of activations from a wide distribution comprised of multiple large components into a narrow one with a single dominant mode. Notice that despite a dominant component emerging, other components do not necessarily vanish rather their contribution diminishes. We will later show that such imbalance between λ_k of various components is not large enough to trigger major mode collapse. These observations confirm that our hypothesis regarding the nature of the underlying distribution is valid and mixture normalization in practice, follows what its formulation is advocating.

[7] It refers to when the fifth batch normalization in the third branch of the first inception layer in second inception block is replaced with mixture normalization.

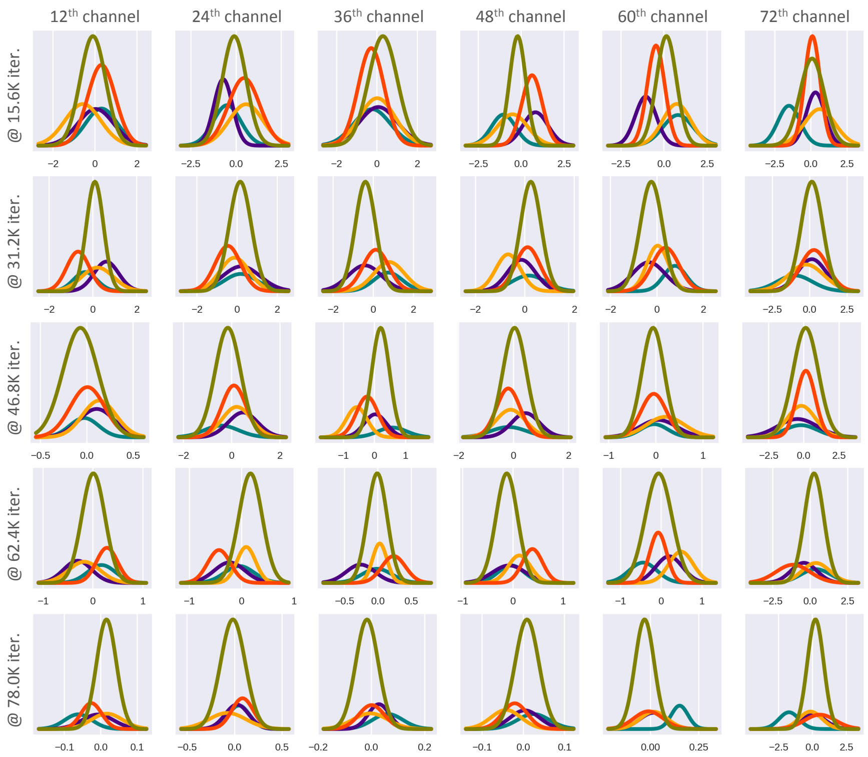


Figure 6.8: Evolution of mixture normalization associated to "inc2/0/2/4" layer as training MN-4 on CIFAR-100 progresses. As argued before, we observe that the underlying distribution is comprised of multiple modes of variation. While these sub-populations are of relatively uniform importance at the beginning, as training procedure goes on, mixture components evolve where some get closer, while others are pushed away from each other creating more distinct components. Here, different colors index mixture components, when sorted according to λ_k values.

6.3.3 Effective Number of Mixture Components

One of the hyperparameters of mixture normalization is the number of its components, K, which must be specified as a choice of design. In some cases, K-means clustering or GMM may generate components with very small λ_k, meaning that the corresponding component is not very representative. Therefore, we, in our implementation, have opted heuristics to discard components whose normalized λ_k is less than 0.01, where samples associated to those are then merged with the remaining components. If the underlying data distribution is sufficiently well approximated using a single Gaussian distribution, which is against our proposal, then we should expect mixture normalization to not utilize all the K components. Figure 6.9 illustrates the actual number of components that mixture normalization has used through the entire training procedure of MN-4 (ref. Table 6.4) on CIFAR-10 and CIFAR-100. We show the curves associated to all the 10 mixture normalization modules in "inc2/0" and "inc3/0". We observe that except two cases, the rest of mixture normalization modules consistently utilize all the K=5 components. This suggests that despite the potential appearance of a dominant component (ref. Section 6.3.2), properly estimating the underlying distribution of activations, still prefers a mixture model over a single Gaussian distribution.

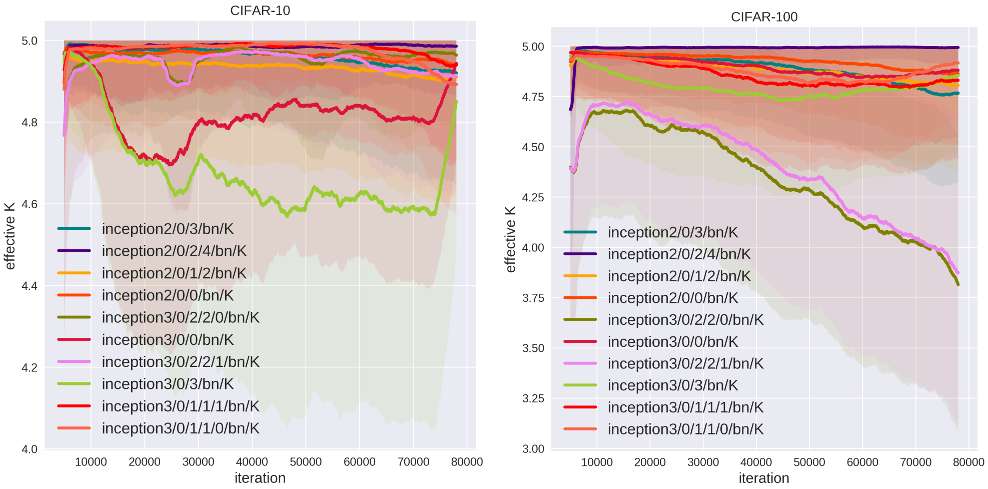


Figure 6.9: Effect of the number of mixture components in MN-4 using Inception-V3 when trained on CIFAR-10 and CIFAR-100. For the sake of better visualization, curves are smoothed using running average and one standard deviation is shown as the shaded area. We can see that the majority of MN modules fully utilize all (K=5) their mixture components, indicating that the need for better approximation using mixture model does not disappear, rather slightly diminishes, as the training procedure continues.

6.4 Summary

In this Chapter, we demonstrated how normalizing transform, employed in batch normalization and its variants, is related to the kernels from generative probability models. We showed that, the distribution of activations associated with internal layers of deep convolutional neural networks, illustrates an asymmetric characteristic and is very likely to be better estimated using a mixture model. We proposed Mixture Normalization (MN), where a Gaussian mixture model initially identifies modes of variation in the underlying distribution, and then each sample in the mini-batch is normalized using mean and standard deviation of the mixture component to which it belongs to. We confirmed through extensive set of experiments on CIFAR-10 and CIFAR-100, that Mixture Normalization not only significantly accelerates convergence of batch normalized models but also achieves better final test accuracy.

CHAPTER 7: CONCLUSION AND FUTURE WORK

Here we highlight the concluding remarks on this dissertation, and expand on potential future work in this direction of research.

7.1 Conclusion

In this dissertation, we addressed the problem of semantically describing images using textual tags and visual attributes. In Chapter 3, we proposed a mathematical framework based on non-negative matrix factorization to perform automatic image annotation such that the proposed technique can seamlessly adapt to the continuous growth of datasets. Our proposed approach can be seen as a query-specific technique which is built on the features of nearest-neighbors and tags. It naturally solves the problem of feature fusion and handles the challenge of rare tags by introducing weight matrices that penalize for incorrect modeling of less frequent tags and images that are associated with them. Despite their effectiveness, tags have an inherent problem. That is, the number of semantic concepts they can represent is identical to the number of labels the model has learned. In contrast, attributes are category agnostic. Hence, their combination can be used to model exponential number of semantic classes. Given the above superiority, in Chapter 4, we focused on visual attributes. We proposed that integrating semantic segmentation in form of semantic face and body parsing improves person-related attribute prediction task. This is being done through exploiting localization cues that corresponding semantic parsing task would provide. Our view is motivated by the fact that most attributes describe local characteristics in images. We evaluated our proposed novel approach on multiple face and full body person attributes and achieved state-of-the-art performance. We also demonstrated that image-level attribute labels can be exploited to boost the semantic parsing task once the number of annotations for training is limited. This indicates that labels for semantic segmentation can be obtained in a significantly cheaper procedure. Next, in Chapter 5, we used both textual tags and visual attributes as semantic descriptors to analyse the large-scale selfie phenomenon from a computer vision and machine learning standpoint. Particularly, we illustrated how the appearance of certain objects or concepts can influence the popularity and sentiment of selfies. Meanwhile, we collected the first of its kind selfie dataset for the research purposes. Finally, in Chapter 6, we provided a fresh view, in light of information geometry, to why the widely-used Batch Normalization layer works. This layer is used in almost all today's state-of-the-art architectures including those that were proposed in this dissertation. Specifically, we gave theoretic grounds on how such a normalization layer can be improved by disentangling modes of variation in the underlying distribution of internal layers. Our experiments confirmed that the proposed mixture normalization effectively accelerates training of different batch-normalized architectures including Inception-V3, DenseNet with 40 and 100 layers, and DCGAN, while achieving better generalization error.

7.2 Future Work

In Chapter 4, we proposed integrating semantic segmentation task with person-related attribute prediction. However, a similar joint learning paradigm can be imagined for any task, which especially relies on detailed localization cues. For instance, fine-grained image classification, or learning fine-grained image similarity are among potential candidates. In addition, problems such as image caption generation or visual question answering ,that require spatial attention to various parts of the image would certainly enjoy semantic localization cues, which semantic segmentation task provides. Furthermore, given that a few video semantic segmentation datasets have recently been introduced, one may extend similar approaches as those that were proposed in this dissertation, from image domain to the video.

Another line of future work is to reduce our reliance on expensive supervised annotations, which semantic segmentation task usually requires. Generating dense pixel-level annotations of relevant categories for a new recognition task is extremely laborious and costly. Instead, we can move from dense to coarse labeling. Particularly, we will have to only label a small portion of pixels. This introduces the research challenge of maintaining the quality of segmentation masks in the above weekly supervised paradigm, so the main recognition task still enjoys a reasonable performance gain.

In Chapter 6, we proposed to accelerate training of batch-normalized deep convolutional neural architecture by disentangling modes of variation in the distribution of layer outputs. We realized it using Gaussian mixture model (GMM). However, the number of mixture components are fixed a priori, as a design choice. This is certainly sub-optimal because not only different layers are of different complexities, but also the associated distributions change as we proceed through the training procedure. In an orthogonal approach, instead of GMM, we can estimate the underlying distribution of layer outputs with Skew-normal distribution. It is close enough to Gaussian density function yet offers sufficient room to model the asymmetry in the probability density function.

LIST OF REFERENCES

[1] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

[2] M. M. Kalayeh, B. Gong, and M. Shah, “Improving facial attribute prediction using semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6942–6950.

[3] V. Le, J. Brandt, Z. Lin, L. Bourdev, and T. S. Huang, “Interactive facial feature localization,” in *European Conference on Computer Vision*. Springer, 2012, pp. 679–692.

[4] B. M. Smith, L. Zhang, J. Brandt, Z. Lin, and J. Yang, “Exemplar-based face parsing,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 3484–3491.

[5] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, “Advances in optimizing recurrent networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, 2013, pp. 8624–8628.

[6] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.

[7] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks.” in *CVPR*, vol. 1, no. 2, 2017, p. 3.

[8] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.

[9] C. Huang, Y. Li, C. Change Loy, and X. Tang, “Learning deep representation for imbalanced classification,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5375–5384.

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

[11] Y. Li, C. Huang, C. C. Loy, and X. Tang, “Human attribute recognition by deep hierarchical contexts,” in *European Conference on Computer Vision*. Springer, 2016, pp. 684–700.

[12] L. Bourdev, S. Maji, and J. Malik, “Describing people: A poselet-based approach to attribute classification,” in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 1543–1550.

[13] A. Makadia, V. Pavlovic, and S. Kumar, “A new baseline for image annotation,” in *European conference on computer vision*. Springer, 2008, pp. 316–329.

[14] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid, “Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation,” in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 309–316.

[15] S. Zhang, J. Huang, Y. Huang, Y. Yu, H. Li, and D. N. Metaxas, “Automatic image annotation using group sparsity,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3312–3319.

[16] M. Chen, A. Zheng, and K. Weinberger, “Fast image tagging,” in *International conference on machine learning*, 2013, pp. 1274–1282.

[17] Y. Verma and C. Jawahar, “Exploring svm for image annotation in presence of confusing labels.” in *BMVC*, 2013, pp. 25–1.

[18] X. Xu, A. Shimada, and R.-i. Taniguchi, “Image annotation by learning label-specific distance metrics,” in *International Conference on Image Analysis and Processing*. Springer, 2013, pp. 101–110.

[19] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Advances in neural information processing systems*, 2001, pp. 556–562.

[20] E. Gaussier and C. Goutte, “Relation between plsa and nmf and implications,” in *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2005, pp. 601–602.

[21] C. Ding, T. Li, and W. Peng, “On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing,” *Computational Statistics & Data Analysis*, vol. 52, no. 8, pp. 3913–3927, 2008.

[22] J. Liu, C. Wang, J. Gao, and J. Han, “Multi-view clustering via joint nonnegative matrix factorization,” in *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 252–260.

[23] K. Gong, X. Liang, D. Zhang, X. Shen, and L. Lin, “Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 932–940.

[24] M. Lin, Q. Chen, and S. Yan, “Network in network,” *arXiv preprint arXiv:1312.4400*, 2013.

[25] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1520–1528.

[26] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the inception architecture for computer vision,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[27] A. Khosla, A. Das Sarma, and R. Hamid, “What makes an image popular?” in *ACM WWW*, 2014.

[28] D. Borth, R. Ji, T. Chen, T. Breuel, and S.-F. Chang, “Large-scale visual sentiment ontology and detectors using adjective noun pairs,” in *ACM MM*, 2013.

[29] S. Bakhshi, D. A. Shamma, and E. Gilbert, “Faces engage us: Photos with faces attract more likes and comments on instagram,” in *ACM CHI*, 2014.

[30] T. Jaakkola and D. Haussler, “Exploiting generative models in discriminative classifiers,” in *Advances in neural information processing systems*, 1999, pp. 487–493.

[31] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, “Inception-v4, inception-resnet and the impact of residual connections on learning.” 2017.

[32] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[33] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.

[34] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis.”

[35] Y. Wu and K. He, “Group normalization,” *arXiv preprint arXiv:1803.08494*, 2018.

[36] M. Ren, R. Liao, R. Urtasun, F. H. Sinz, and R. S. Zemel, "Normalizing the normalizers: Comparing and extending network normalization schemes," *arXiv preprint arXiv:1611.04520*, 2016.

[37] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[38] D. M. Titterington, A. F. Smith, and U. E. Makov, *Statistical analysis of finite mixture distributions*. Wiley,, 1985.

[39] S. Ioffe, "Batch renormalization: Towards reducing minibatch dependence in batch-normalized models," in *Advances in Neural Information Processing Systems*, 2017, pp. 1942–1950.

[40] K. Barnard, P. Duygulu, D. Forsyth, N. d. Freitas, D. M. Blei, and M. I. Jordan, "Matching words and pictures," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1107–1135, 2003.

[41] F. Monay and D. Gatica-Perez, "Plsa-based image auto-annotation: constraining the latent space," in *Proceedings of the 12th annual ACM international conference on Multimedia*. ACM, 2004, pp. 348–351.

[42] O. Yakhnenko and V. Honavar, "Annotating images and image objects using a hierarchical dirichlet process model," in *Proceedings of the 9th International Workshop on Multimedia Data Mining: held in conjunction with the ACM SIGKDD 2008*. ACM, 2008, pp. 1–7.

[43] P. Duygulu, K. Barnard, J. F. de Freitas, and D. A. Forsyth, "Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary," in *European conference on computer vision*. Springer, 2002, pp. 97–112.

[44] R. Socher and L. Fei-Fei, “Connecting modalities: Semi-supervised segmentation and annotation of images using unaligned text corpora,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 966–973.

[45] Y. Xiang, X. Zhou, T.-S. Chua, and C.-W. Ngo, “A revisit of generative model for automatic image annotation using markov random fields,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1153–1160.

[46] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos, “Supervised learning of semantic classes for image annotation and retrieval,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 3, pp. 394–410, 2007.

[47] S. Feng, R. Manmatha, and V. Lavrenko, “Multiple bernoulli relevance models for image and video annotation,” in *null*. IEEE, 2004, pp. 1002–1009.

[48] A. Yavlinsky, E. Schofield, and S. Rüger, “Automated image annotation using global features and robust nonparametric density estimation,” in *International Conference on Image and Video Retrieval*. Springer, 2005, pp. 507–517.

[49] C. Cusano, G. Ciocca, and R. Schettini, “Image annotation using svm,” in *Internet imaging V*, vol. 5304. International Society for Optics and Photonics, 2003, pp. 330–339.

[50] D. Grangier and S. Bengio, “A discriminative kernel-based model to rank images from text queries,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 10, no. LIDIAP-ARTICLE-2008-010, 2008.

[51] T. Hertz, A. Bar-Hillel, and D. Weinshall, “Learning distance functions for image retrieval,” in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2. IEEE, 2004, pp. II–II.

[52] Y. Verma and C. Jawahar, "Image annotation using metric learning in semantic neighbourhoods," in *European Conference on Computer Vision*. Springer, 2012, pp. 836–849.

[53] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 2003, pp. 267–273.

[54] D. Guillamet, B. Schiele, and J. Vitria, "Analyzing non-negative matrix factorization for image classification," in *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, vol. 2. IEEE, 2002, pp. 116–119.

[55] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, p. 788, 1999.

[56] D. Greene and P. Cunningham, "A matrix factorization approach for integrating multiple data views," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2009, pp. 423–438.

[57] M. M. Kalayeh, H. Idrees, and M. Shah, "Nmf-knn: Image annotation using weighted multi-view non-negative matrix factorization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 184–191.

[58] M. Yang, L. Zhang, D. Zhang, and S. Wang, "Relaxed collaborative representation for pattern classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2224–2231.

[59] D. Guillamet, M. Bressan, and J. Vitria, "A weighted non-negative matrix factorization for local representations," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–I.

[60] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth, “Describing objects by their attributes,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 1778–1785.

[61] C. H. Lampert, H. Nickisch, and S. Harmeling, “Learning to detect unseen object classes by between-class attribute transfer,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 951–958.

[62] A. Farhadi, I. Endres, and D. Hoiem, “Attribute-centric recognition for cross-category generalization,” in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2352–2359.

[63] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar, “Attribute and simile classifiers for face verification,” in *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 365–372.

[64] N. Kumar, P. Belhumeur, and S. Nayar, “Facetracer: A search engine for large collections of images with faces,” in *European conference on computer vision*. Springer, 2008, pp. 340–353.

[65] J. Liu, B. Kuipers, and S. Savarese, “Recognizing human actions by attributes,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3337–3344.

[66] T. Berg and P. Belhumeur, “Poof: Part-based one-vs.-one features for fine-grained categorization, face verification, and attribute estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 955–962.

[67] H. Chen, A. Gallagher, and B. Girod, “Describing clothing by semantic attributes,” in *European conference on computer vision*. Springer, 2012, pp. 609–623.

[68] S. J. Hwang, F. Sha, and K. Grauman, “Sharing features between objects and their attributes,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1761–1768.

[69] D. Jayaraman, F. Sha, and K. Grauman, “Decorrelating semantic visual attributes by resisting the urge to share,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1629–1636.

[70] A. Vedaldi, S. Mahendran, S. Tsogkas, S. Maji, R. Girshick, J. Kannala, E. Rahtu, I. Kokkinos, M. B. Blaschko, D. Weiss *et al.*, “Understanding objects in detail with fine-grained attributes,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3622–3629.

[71] Y. Wang and G. Mori, “A discriminative latent model of object classes and attributes,” in *European Conference on Computer Vision*. Springer, 2010, pp. 155–168.

[72] D. Parikh and K. Grauman, “Interactively building a discriminative vocabulary of nameable attributes,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1681–1688.

[73] C. Gan, T. Yang, and B. Gong, “Learning attributes equals multi-source domain generalization,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 87–97.

[74] G. Gkioxari, R. Girshick, and J. Malik, “Actions and attributes from wholes and parts,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2470–2478.

[75] N. Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. Bourdev, "Panda: Pose aligned networks for deep attribute modeling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1637–1644.

[76] K. K. Singh and Y. J. Lee, "End-to-end localization and ranking for relative attributes," in *European Conference on Computer Vision*. Springer, 2016, pp. 753–769.

[77] M. Jaderberg, K. Simonyan, A. Zisserman *et al.*, "Spatial transformer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2017–2025.

[78] E. Rudd, M. Günther, and T. Boult, "Moon: A mixed objective optimization network for the recognition of facial attributes," *arXiv preprint arXiv:1603.07027*, 2016.

[79] Q. Dong, S. Gong, and X. Zhu, "Class rectification hard mining for imbalanced deep learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1851–1860.

[80] J. Li, F. Zhao, J. Feng, S. Roy, S. Yan, and T. Sim, "Landmark free face attribute prediction," *IEEE Transactions on Image Processing*, vol. 27, no. 9, pp. 4651–4662, 2018.

[81] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *Computer vision and pattern recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[82] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore, "Real-time human pose recognition in parts from single depth images," *Communications of the ACM*, vol. 56, no. 1, pp. 116–124, 2013.

[83] D. Ciresan, A. Giusti, L. M. Gambardella, and J. Schmidhuber, "Deep neural networks segment neuronal membranes in electron microscopy images," in *Advances in neural information processing systems*, 2012, pp. 2843–2851.

[84] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[85] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[86] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” *arXiv preprint arXiv:1511.07122*, 2015.

[87] 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*, vol. 40, no. 4, pp. 834–848, 2018.

[88] C. Liang-Chieh, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected crfs,” in *International Conference on Learning Representations*, 2015.

[89] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[90] G. Lin, A. Milan, C. Shen, and I. Reid, “Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation,” *arXiv preprint arXiv:1611.06612*, 2016.

[91] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” *arXiv preprint arXiv:1706.05587*, 2017.

[92] A. Kae, K. Sohn, H. Lee, and E. Learned-Miller, “Augmenting crfs with boltzmann machine shape priors for image labeling,” in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 2019–2026.

[93] S. Liu, J. Yang, C. Huang, and M.-H. Yang, “Multi-objective convolutional learning for face labeling,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3451–3459.

[94] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille, “Detect what you can: Detecting and representing objects using holistic models and body parts,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[95] P. Wang, X. Shen, Z. Lin, S. Cohen, B. Price, and A. L. Yuille, “Joint object and part segmentation using deep learned potentials,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1573–1581.

[96] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, “Hypercolumns for object segmentation and fine-grained localization,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 447–456.

[97] X. Liang, X. Shen, J. Feng, L. Lin, and S. Yan, “Semantic object parsing with graph lstm,” in *European Conference on Computer Vision*. Springer, 2016, pp. 125–143.

[98] L.-C. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille, “Attention to scale: Scale-aware semantic image segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3640–3649.

[99] F. Xia, P. Wang, L.-C. Chen, and A. L. Yuille, “Zoom better to see clearer: Human and object parsing with hierarchical auto-zoom net,” in *European Conference on Computer Vision*. Springer, 2016, pp. 648–663.

[100] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg, “Parsing clothing in fashion photographs,” in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3570–3577.

[101] X. Liang, S. Liu, X. Shen, J. Yang, L. Liu, J. Dong, L. Lin, and S. Yan, “Deep human parsing with active template regression,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 12, pp. 2402–2414, 2015.

[102] X. Liang, C. Xu, X. Shen, J. Yang, S. Liu, J. Tang, L. Lin, and S. Yan, “Human parsing with contextualized convolutional neural network,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1386–1394.

[103] J. Dong, Q. Chen, W. Xia, Z. Huang, and S. Yan, “A deformable mixture parsing model with parselets,” in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3408–3415.

[104] S. Liu, X. Liang, L. Liu, K. Lu, L. Lin, X. Cao, and S. Yan, “Fashion parsing with video context,” *IEEE Transactions on Multimedia*, vol. 17, no. 8, pp. 1347–1358, 2015.

[105] W. Yang, P. Luo, and L. Lin, “Clothing co-parsing by joint image segmentation and labeling,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 3182–3189.

[106] K. Yamaguchi, M. H. Kiapour, and T. L. Berg, “Paper doll parsing: Retrieving similar styles to parse clothing items,” in *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3519–3526.

[107] S. Liu, X. Liang, L. Liu, X. Shen, J. Yang, C. Xu, L. Lin, X. Cao, and S. Yan, “Matching-cnn meets knn: Quasi-parametric human parsing,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1419–1427.

[108] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, “On large-batch training for deep learning: Generalization gap and sharp minima,” *arXiv preprint arXiv:1609.04836*, 2016.

[109] E. Hoffer, I. Hubara, and D. Soudry, “Train longer, generalize better: closing the generalization gap in large batch training of neural networks,” in *Advances in Neural Information Processing Systems*, 2017, pp. 1731–1741.

[110] T. Salimans and D. P. Kingma, “Weight normalization: A simple reparameterization to accelerate training of deep neural networks,” in *Advances in Neural Information Processing Systems*, 2016, pp. 901–909.

[111] L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, 2016, pp. 2414–2423.

[112] M. A. Figueiredo, R. D. Nowak, and S. J. Wright, “Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems,” *IEEE Journal of selected topics in signal processing*, vol. 1, no. 4, pp. 586–597, 2007.

[113] L. Von Ahn and L. Dabbish, “Labeling images with a computer game,” in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2004, pp. 319–326.

[114] A. Oliva and A. Torralba, “Modeling the shape of the scene: A holistic representation of the spatial envelope,” *International journal of computer vision*, vol. 42, no. 3, pp. 145–175, 2001.

[115] J. Van De Weijer and C. Schmid, “Coloring local feature extraction,” in *European conference on computer vision*. Springer, 2006, pp. 334–348.

[116] V. Lavrenko, R. Manmatha, and J. Jeon, "A model for learning the semantics of pictures," in *Advances in neural information processing systems*, 2004, pp. 553–560.

[117] D. Metzler and R. Manmatha, "An inference network approach to image retrieval," in *International Conference on Image and Video Retrieval*. Springer, 2004, pp. 42–50.

[118] J. Liu, M. Li, Q. Liu, H. Lu, and S. Ma, "Image annotation via graph learning," *Pattern recognition*, vol. 42, no. 2, pp. 218–228, 2009.

[119] H. Fu, Q. Zhang, and G. Qiu, "Random forest for image annotation," in *ECCV*, 2012.

[120] M. Muja and D. G. Lowe, "Fast matching of binary features," in *Computer and Robot Vision (CRV), 2012 Ninth Conference on*. IEEE, 2012, pp. 404–410.

[121] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[122] H. Bilen and A. Vedaldi, "Weakly supervised deep detection networks," in *CVPR*, 2016.

[123] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[124] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[125] Y. Xiong, K. Zhu, D. Lin, and X. Tang, "Recognize complex events from static images by fusing deep channels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1600–1609.

[126] J. Wang, Y. Cheng, and R. Schmidt Feris, "Walk and learn: Facial attribute representation learning from egocentric video and contextual data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2295–2304.

[127] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *European Conference on Computer Vision*. Springer, 2014, pp. 346–361.

[128] Y. Zhong, J. Sullivan, and H. Li, "Leveraging mid-level deep representations for predicting face attributes in the wild," in *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 3239–3243.

[129] N. Sarafianos, X. Xu, and I. A. Kakadiaris, "Deep imbalanced attribute classification using visual attention aggregation," in *The European Conference on Computer Vision (ECCV)*, September 2018.

[130] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[131] G. Gkioxari, R. Girshick, and J. Malik, "Contextual action recognition with r* cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1080–1088.

[132] M. S. Sarfraz, A. Schumann, Y. Wang, and R. Stiefelhagen, "Deep view-sensitive pedestrian attribute inference in an end-to-end model," *arXiv preprint arXiv:1707.06089*, 2017.

[133] F. Zhu, H. Li, W. Ouyang, N. Yu, and X. Wang, "Learning spatial regularization with image-level supervisions for multi-label image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5513–5522.

[134] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, “Imagenet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[135] M. M. Kalayeh, M. Seifu, W. LaLanne, and M. Shah, “How to take a good selfie?” in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 923–926.

[136] H. Jégou, M. Douze, C. Schmid, and P. Pérez, “Aggregating local descriptors into a compact image representation,” in *CVPR*, 2010, pp. 3304–3311.

[137] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.

[138] 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,” 2014.

[139] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, “Overfeat: Integrated recognition, localization and detection using convolutional networks,” in *ICLR*, 2014.

[140] M. M. Kalayeh and M. Shah, “Training faster by separating modes of variation in batch-normalized models,” *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[141] S.-i. Amari and H. Nagaoka, *Methods of information geometry*. American Mathematical Soc., 2007, vol. 191.

[142] S.-I. Amari, “Natural gradient works efficiently in learning,” *Neural computation*, vol. 10, no. 2, pp. 251–276, 1998.

[143] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek, "Image classification with the fisher vector: Theory and practice," *International journal of computer vision*, vol. 105, no. 3, pp. 222–245, 2013.

[144] M. I. Jordan and R. A. Jacobs, "Hierarchical mixtures of experts and the em algorithm," *Neural computation*, vol. 6, no. 2, pp. 181–214, 1994.

[145] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[146] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[147] 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)*, pp. 1–38, 1977.

[148] R. Hosseini and S. Sra, "An alternative to em for gaussian mixture models: Batch and stochastic riemannian optimization," *arXiv preprint arXiv:1706.03267*, 2017.

[149] A. Krizhevsky, "Learning multiple layers of features from tiny images," 2009.

[150] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[151] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[152] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger, "Deep networks with stochastic depth," in *European Conference on Computer Vision*. Springer, 2016, pp. 646–661.

[153] M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," in *International Conference on Machine Learning*, 2016, pp. 1225–1234.

[154] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang, "Stronger generalization bounds for deep nets via a compression approach," *arXiv preprint arXiv:1802.05296*, 2018.

[155] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang *et al.*, "Photo-realistic single image super-resolution using a generative adversarial network," *arXiv preprint*, 2016.

[156] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1125–1134.

[157] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[158] M. Arjovsky and L. Bottou, "Towards principled methods for training generative adversarial networks," *arXiv preprint arXiv:1701.04862*, 2017.

[159] T. Che, Y. Li, A. P. Jacob, Y. Bengio, and W. Li, "Mode regularized generative adversarial networks," *arXiv preprint arXiv:1612.02136*, 2016.

[160] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2016, pp. 2172–2180.

[161] L. Metz, B. Poole, D. Pfau, and J. Sohl-Dickstein, "Unrolled generative adversarial networks," *arXiv preprint arXiv:1611.02163*, 2016.

[162] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[163] A. Ghosh, V. Kulharia, V. Namboodiri, P. H. Torr, and P. K. Dokania, "Multi-agent diverse generative adversarial networks," *arXiv preprint arXiv:1704.02906*, 2017.

[164] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *Proceedings of the 3rd International Conference on Learning Representations (ICLR) arXiv:1412.6980*, 2015.

[165] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Advances in Neural Information Processing Systems*, 2017, pp. 6629–6640.

[166] S. Tokui, K. Oono, S. Hido, and J. Clayton, "Chainer: a next-generation open source framework for deep learning," in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)*, 2015. [Online]. Available: http://learningsys.org/papers/LearningSys_2015_paper_33.pdf

[167] R. Okuta, Y. Unno, D. Nishino, S. Hido, and C. Loomis, "Cupy: A numpy-compatible library for nvidia gpu calculations," in *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, 2017. [Online]. Available: http://learningsys.org/nips17/assets/papers/paper_16.pdf

[168] O. Bachem, M. Lucic, S. H. Hassani, and A. Krause, "Approximate k-means++ in sublinear time." 2016.

[169] O. Bachem, M. Lucic, H. Hassani, and A. Krause, “Fast and provably good seedings for k-means,” in *Advances in Neural Information Processing Systems*, 2016, pp. 55–63.

[170] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, “Scalable k-means++,” *Proceedings of the VLDB Endowment*, vol. 5, no. 7, pp. 622–633, 2012.

[171] O. Bachem, M. Lucic, and A. Krause, “Distributed and provably good seedings for k-means in constant rounds,” in *International Conference on Machine Learning*, 2017, pp. 292–300.

[172] J. J. Verbeek, N. Vlassis, and B. Kröse, “Efficient greedy learning of gaussian mixture models,” *Neural computation*, vol. 15, no. 2, pp. 469–485, 2003.

[173] M. Lucic, M. Faulkner, A. Krause, and D. Feldman, “Training gaussian mixture models at scale via coresets,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 5885–5909, 2017.