Evaluating and Interpreting Deep Convolutional Neural Networks via Non-negative Matrix Factorization

Thèse N° 9372

Présentée le 21 juin 2019

à la Faculté informatique et communications
Laboratoire d'images et représentation visuelle
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Edo COLLINS

Acceptée sur proposition du jury

Prof. M. Jaggi, président du jury
Prof. S. Süsstrunk, directrice de thèse
Prof. R. Sznitman, rapporteur
Dr H. Ben-Shitrit, rapporteur
Dr M. Salzmann, rapporteur

2019

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

It's a dangerous business, Frodo,
going out your door.
You step onto the road,
and if you don't keep your feet,
there's no knowing
where you might be swept off to.
— Bilbo Baggins

To Tuli the cat…

Acknowledgements

The last years at EPFL were a time of great change for me, both professionally and personally. Now at the high point of this rollercoaster ride, I feel very fortunate to have had such good people around me, whose support carried me through the twists and the turns.

First, to my thesis advisor, Prof. Sabine Süsstrunk - thank you. I am grateful for the opportunity you gave me, the patience and the encouragement. The confidence you put in me and your sound advice were invaluable. I am looking forward to the next chapter in our collaboration!
It was a great honor to present before the jury of my thesis committee: Prof. Martin Jaggi, Prof. Raphael Sznitman, Dr. Mathieu Salzmann and Dr. Horesh Ben-Shitrit.

I would like to thank my colleagues at IVRL. Great thanks to Marjan Shahpaski for always interesting, always encouraging office conversations - we both would have graduated a year sooner without them, but it was time well spent. I would like to thank the next door neighbours, Majed El Helou and Fayez Lahoud, for great talks and great feedback. A special thanks to Dr. Radhakrishna Achanta for helping me start on the path that would ultimately lead to this thesis, and to Dr. Siavash Arjomand Bigdeli for helping me see it through. Thanks also to Ruofan Zhou and Sami Arpa.
A great thank you goes to Francoise Behn, without whom Swiss bureaucracy would have claimed another victim.

To the wonderful friends I have been so lucky to have: Yahel, Assaf, Uri, Ido, Anni, Stuart, Winston, Benoit, Ya-Ping, Elodie - thank you for keeping me sane, and keeping me happy. Last but not least, my family, Rachel, Miri, Judy and Jacob - as I hope you already know - whatever success I have in my life is thanks to your love and support.

This thesis is dedicated to all of you (not just the cat).

Lausanne, May 2019

Abstract

With ever greater computational resources and more accessible software, deep neural networks have become ubiquitous across industry and academia. Their remarkable ability to generalize to new samples defies the conventional view, which holds that complex, over-parameterized networks would be prone to overfitting. This apparent discrepancy is exacerbated by our inability to inspect and interpret the high-dimensional, non-linear, latent representations they learn, which has led many to refer to neural networks as "black-boxes". The Law of Parsimony states that "simpler solutions are more likely to be correct than complex ones". Since they perform quite well in practice, a natural question to ask, then, is *in what way are neural networks simple?*

We propose that *compression* is the answer. Since good generalization requires invariance to irrelevant variations in the input, it is necessary for a network to discard this irrelevant information. As a result, semantically similar samples are mapped to similar representations in neural network *deep feature space*, where they form simple, low-dimensional structures. Conversely, a network that overfits relies on memorizing individual samples. Such a network cannot discard information as easily.

In this thesis we characterize the difference between such networks using the *non-negative rank* of activation matrices. Relying on the non-negativity of rectified-linear units, the non-negative rank is the smallest number that admits an exact *non-negative matrix factorization*. We derive an upper bound on the amount of memorization in terms of the non-negative rank, and show it is a natural complexity measure for rectified-linear units.

We use *approximate* non-negative matrix factorization to show compression can be successfully used to distinguish between networks with different levels of memorization and generalization. This observation is confirmed over several datasets and network architectures, and we show that it even holds at the level of individual output classes, as well as during training.

With a focus on deep convolutional neural networks trained to perform object recognition, we show that the two non-negative factors derived from deep network layers decompose the information held therein in an interpretable way. The first of these factors provides *heatmaps* which highlight similarly encoded regions within an input image or image set. Shedding some light into the "black box", these heatmaps reveal what information a network finds relevant. We find that these networks learn to detect *semantic parts* and form a hierarchy, such that

parts are further broken down into sub-parts. We quantitatively evaluate the semantic quality of these heatmaps by using them to perform semantic co-segmentation and co-localization. In spite of the convolutional network we use being trained solely with image-level labels, we achieve results comparable or better than domain-specific state-of-the-art methods for these tasks.
The second non-negative factor provides a *bag-of-concepts* representation for an image or image set. We use this representation to derive global image descriptors for images in a large collection. With these descriptors in hand, we perform two variations content-based image retrieval, i.e. reverse image search. Using information from one of the non-negative matrix factors we obtain descriptors which are suitable for finding semantically related images, i.e., belonging to the same semantic category as the query image. Combining information from both non-negative factors, however, yields descriptors that are suitable for finding other images of the specific instance depicted in the query image, where we again achieve state-of-the-art performance.

Keywords: deep neural networks, convolutional neural networks, non-negative matrix factorization, generalization, overfitting, network interpretability, co-segmentation, co-localization, content-based image retrieval

Résumé

Avec des ressources informatiques de plus en plus importantes et des logiciels plus accessibles, les réseaux neuronaux profonds sont devenus omniprésents dans l'industrie et le milieu universitaire. Leur remarquable capacité de généralisation à de nouveaux échantillons défie l'opinion conventionnelle selon laquelle des réseaux complexes et sur-paramétrés seraient susceptibles de surapprentissage. Cette contradiction apparente est exacerbée par notre incapacité d'inspecter et d'interpréter les représentations latentes, non linéaires et hautement dimensionnelles que ces réseaux apprennent, ce qui a amené beaucoup à qualifier les réseaux neuronaux de "boîtes noires". La loi de la parcimonie stipule que "des solutions plus simples ont plus de chances d'être correctes que des solutions complexes". Puisqu'ils fonctionnent assez bien dans la pratique, une question naturelle à se poser est donc de savoir en quoi les réseaux de neurones sont simples ?

Nous proposons que la compression est la réponse. Étant donné qu'une bonne généralisation exige l'invariance à des variations non pertinentes de l'entrée, il est nécessaire qu'un réseau se débarrasse de cette information non pertinente. Par conséquent, des échantillons sémantiquement similaires sont mappés à des représentations similaires dans l'espace de charactères profonds d'un réseau neuronal, où ils forment des structures simples et de faible dimension. Inversement, un réseau surapprend s'il mémorise des échantillons individuels. Un tel réseau ne peut pas se débarrasser de l'information aussi facilement.

Dans cette thèse, nous caractérisons la différence entre de tels réseaux en utilisant le *rang non-négatif* des matrices d'activation. En se basant sur la non-négativité des unités linéaires rectifiées, le rang non-négatif est le plus petit nombre qui admet une factorisation exacte de la matrice non-négative. Nous établissons une limite supérieure à la quantité de mémorisation en termes de rang non-négatif, et nous montrons qu'il s'agit d'une mesure de complexité naturelle pour les unités rectifiées linéaires.

Nous utilisons la factorisation matricielle non-négative approximée pour montrer que la compression peut être utilisée avec succès pour distinguer les réseaux avec différents niveaux de mémorisation et de généralisation. Cette observation est confirmée sur plusieurs ensembles de données et architectures de réseau, et nous montrons qu'elle se vérifie même au niveau des classes de sortie individuelles, ainsi qu'au niveau de la formation.

En mettant l'accent sur les réseaux neuronaux convolutionnels profonds entraînés à la reconnaissance d'objets, nous montrons que les deux facteurs non-négatifs dérivés des couches

profondes du réseau décomposent l'information qui s'y trouve d'une manière intelligible. Le premier de ces facteurs fournit des *heat maps* qui mettent en évidence des régions codées de manière similaire dans une image d'entrée ou un ensemble d'images. En éclairant la "boîte noire", ces heat maps révèlent l'information qu'un réseau trouve pertinente. Nous constatons que ces réseaux apprennent à détecter les constituents sèmantiques et à former une hiérarchie, de sorte que les constituants se décomposent davantage en sous-constituants. Nous évaluons quantitativement la qualité sémantique de ces heat maps en les utilisant pour effectuer la co-segmentation et la co-localisation sémantiques. Malgré le réseau convolutif que nous utilisons et le fait que nous ne sommes formés qu'avec des étiquettes de niveau image, nous obtenons pour ces tâches des résultats comparables ou meilleurs que les méthodes qui sont à l'état de l'art du domaine spécifique

Le second facteur non-négatif fournit une représentation de "sac de concepts" pour une image ou un ensemble d'images. Nous utilisons cette représentation pour dériver des descripteurs d'image globaux pour les images d'une grande collection. Avec ces descripteurs en main, nous effectuons deux variantes de la recherche d'images basée sur le contenu, c'est-à-dire la recherche d'images inversée. En utilisant l'information provenant de l'un des facteurs matriciels non-négatifs, nous obtenons des descripteurs qui conviennent à la recherche d'images sémantiquement apparentées, c'est-à-dire appartenant à la même catégorie sémantique que l'image de requête. En combinant les informations des deux facteurs non-négatifs, on obtient des descripteurs qui permettent de trouver d'autres images de l'instance spécifique qui est représentée dans l'image de la requête, où l'on obtient à nouveau des performances de pointe.

Keywords: deep neural networks, convolutional neural networks, non-negative matrix factorization, generalization, overfitting, network interpretability, co-segmentation, co-localization, content-based image retrieval

Contents

List of Figures

List of Tables

Abbreviations and Notation

List of Abbreviations

Abbreviation	Description
NN	Artificial neural network
CNN	Convolutional neural network
SGD	Stochastic gradient descent
PCA	Principal component analysis
NMF	Non-negative matrix factorization
RA	Random ablations
ReLU	Rectified-linear unit
BN	Batch normalization

List of Symbols

Symbol	Description
$\mathbb{R}_+$	Non-negative real numbers [0, inf)
M, P	arbitrary integers representing high dimensional spaces, e.g. $\mathbb{R}_+^M$
$\Lambda(\cdot)$	A function representing a neural network
$\Lambda_i(\cdot)$	A function representing the ith layer of a neural network
ℓ	A loss function defining an optimization objective, not to be confised with
ℓ_2	The Eculidean norm
$\mathsf{X}, \boldsymbol{X}, \boldsymbol{x}$	Neural network input in tensor, matrix and vector form, respectively
I	An input image
A, $\boldsymbol{A}$, $\boldsymbol{a}$	Neural network activations in tensor, matrix and vector form, respectively
$\hat{\mathsf{A}}, \hat{\boldsymbol{A}}, \hat{\boldsymbol{a}}$	Approximation of activations via matrix factorization
X, Z	Random variables representing NN input and hidden activation respectively
$\hat{\boldsymbol{y}}$	NN last layer output
y	Ground truth target NN should predict
Y	A random variable distributed over ground truth targets
$X \times Y$	The true data distribution of input output pairs
A, B	Generic random variables
$\mathcal{P}, \mathcal{I}, \mathcal{C}, \mathcal{H}$	The probability operator, mutual information, common information, and Shannon entropy
$\boldsymbol{V}$	A matrix holding basis vectors of a matrix factorization
$\boldsymbol{U}$	A matrix holding per-sample coefficients for matrix factorization
N	Number of samples in a dataset or batch
H	Height of image or feature map
W	Width of image or feature map
C	Number of channels at a convolutional layer
K	Positive integer rank of matrix factorization, or a random variable defining a categorical distribution on matrix factors
T	Number of output classes
c, k, t	A specific channel, factor, or target, i.e., $1 \leq c \leq C$, $1 \leq k \leq K$, $1 \leq t \leq T$
F	A random variable defining a categorical distribution over each of the K matrix factors
a, b, i, j	Genetic subscripts for enumerating collections
$\mathcal{D}$, $\mathcal{D}_{\text{test}}$	Datasets used for training and testing, respectively
$\mathbf{i}$	A discrete random variable indexing a batch or dataset with $\mathcal{P}(\mathbf{i} = i) = \frac{1}{N}$
p	The probability of randomizing a training label
supp	The *support* of a matrix
rc	The rectangle cover number of a binary matrix

Symbol	Description
U	The matrix $\boldsymbol{U}$ reshaped and rescaled to form a tensor of heatmaps
B	The tensor U after binarization
G	A tensor containing ground truth binary segmentation masks. Unlike y, this is not used for NN training.
R	Number of relevant images with respect to a query image
r	The number of relevant images retrieved up to a given rank
$\boldsymbol{v}$	NMF global image descriptor

1 Introduction

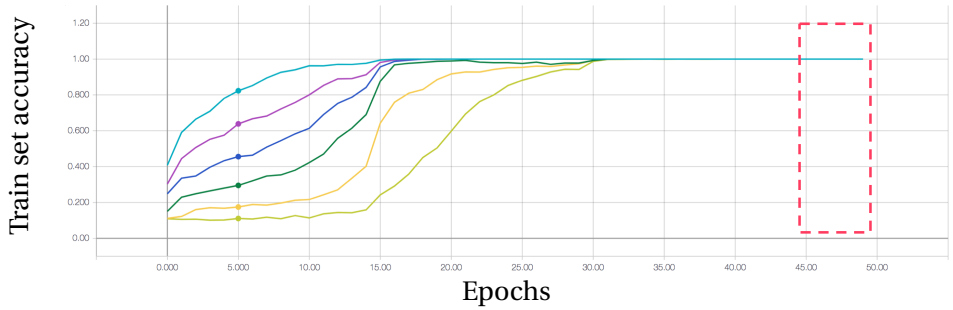
Since the advent of digital computers in the 1940s, researchers have sought to use them to automate tasks traditionally solved by human labor. As computational resources grew, so did the desire to automate more complex and high-level tasks, such as translating natural language texts, recognizing objects in images, etc.

Early strides towards that goal were based on rigid rule-based systems, distilling expert knowledge of a task-specific domain. These systems, however, did not scale to the level of *variation* which exists in most real-world inputs. A task as simple as, for instance, recognizing *dog* images requires approximating a function which maps thousands or even millions of pixels into a yes/no decision, while considering that dogs can be large or small, light or dark, 4-legged or 3-legged, indoors or outdoors - but are not to be confused with cats or even wolves. For such problems, *machine learning* proposed to design algorithms that automatically *learn* how to solve a given task.

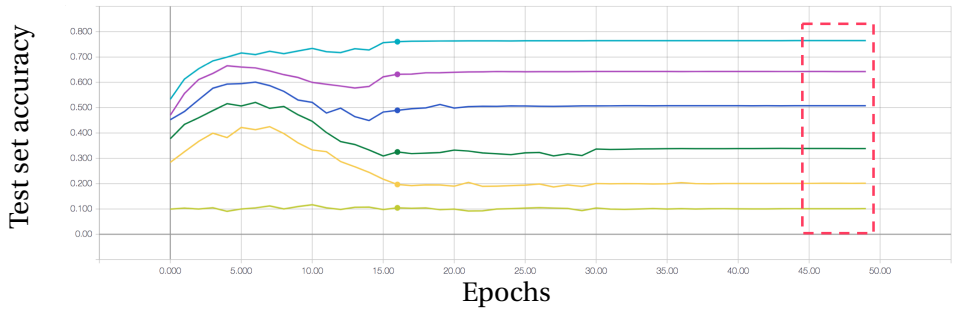
For example, the most common paradigm in machine learning is *supervised learning*, i.e., learning by example. In this setting, a dataset is given which consists of input-output pairs, and a parametric model is trained (i.e., its parameters are gradually refined) to predict the output given the input.

The availability of large datasets and powerful computational resources over the past decade has facilitated the rise in popularity of *deep neural networks*. This flexible class of models is at once general, i.e., requiring little domain-specific expertise, while nonetheless showing state-of-the-art performance across diverse domains. This technology has had a profound impact on the high-tech industry, with the *deep learning* market expected to reach $US 18 Billion by 2024[1].

[1] https://www.marketresearchengine.com/deep-learning-market



(a) Training curves



(b) Test curves

Figure 1.1 – Shown here are (a) training and (b) test curves during training of six convolutional neural networks on the CIFAR-10 dataset forced into different degrees of memorization. Marked in red is the outcome at the end of the training procedure. *Given the final training set accuracy, is it possible to predict the test set accuracy?* In this thesis we find that a distinguishing property which characterizes the difference between these networks, and correlates with better generalization and less memorization, is the *non-negative rank* of their activation matrices.

However, the success of deep networks has also been accompanied by a certain measure of obfuscation. The factors determining the success or failure of learning are not well understood, and methods of evaluating what was learned are limited.

For instance, while the ultimate goal is to *generalize* well to new data, which is not seen during training, some models are prone to *overfitting*, i.e., simply memorizing the training data. In spite of having millions and even billions of parameters, deep neural networks have shown a measure of resilience to this phenomenon, defying the traditional "wisdom" that models with many parameters are likely to overfit. The factors governing a network's tendency to overfit are still subject to intensive study these days.

Consider for instance the learning curves in Figure 1.1. Although all networks achieve perfect accuracy on their training sets, they have very different levels of generalization. What is the distinguishing factor between these networks? Can it be detected independently of a test set?

In this thesis we propose that *compression* is a key factor, distinguishing between networks that generalize well and those that simply *memorize* their training data. Any predictive rule with a degree of generality *must* ignore certain aspects of the instance to which it is applied, effectively compressing it by discarding irrelevant information.

The notion that compression is a hallmark of well-generalizing models is not new. It is an extension of a principle going back to antiquity and made famous as Occam's razor: "Entities are not to be multiplied beyond necessity". In this case, we prefer a network that learns one general rule that accommodates many inputs, over a network that learns many specific rules that accommodate individual inputs.

Compression not only gives us a criterion by which to evaluate generalization behavior, it also addresses questions regarding the *interpretation* of neural networks. It is natural to ask *why* a certain prediction was made, by which reasoning, or by considering which aspects of the input. The nonlinear interactions between the millions of parameters makes exact answers to these questions incomprehensible to human beings. As these networks are deployed into products and services all around us - from targeted advertisement on social media to autonomous vehicles - there is a growing need to validate and understand the outcome of learning.

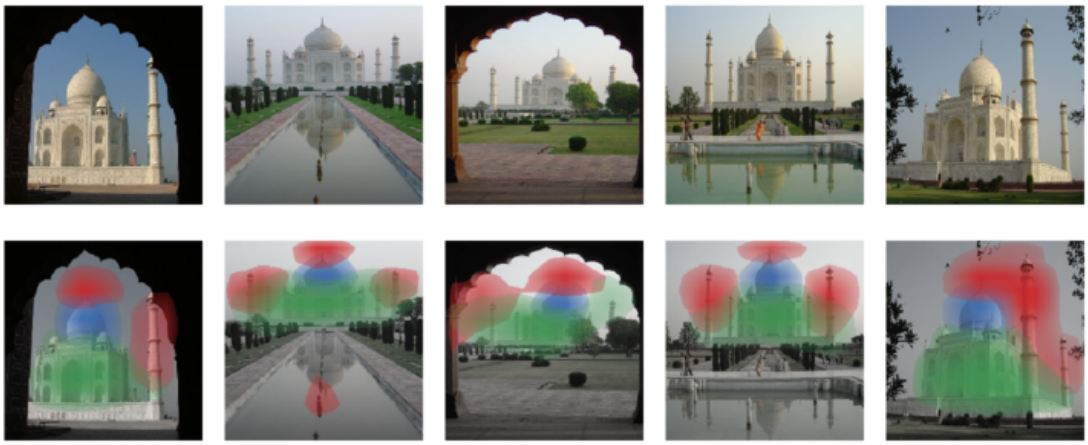
By examining the compression a convolutional neural network applies to images, we can create *heatmaps* which allow us to see how a deep neural network conceptualizes its input. In Figure 1.2 we show an example of this visualization, showing that in a deep network layer, a scene or object is decomposed into its constituent parts. This demonstrates how the network discards information it finds irrelevant, such as scale, perspective, various object deformations etc. At the same time, for the object recognition task it was trained to accomplish, the presence or absence of specific entities, e.g., *tower* or *person*, are deemed important enough to retain.

Furthermore, by characterizing images using the information a convolutional neural network considers relevant about them, we can *semantically* compare two images. This gives a mechanism with which to *search* for semantically similar images given a query image, and *localize* the semantically matching regions within the images.

1.1 Thesis contributions and outline

Our main contribution is actualizing the idea of using compression into a working algorithm. Our tool of choice is *non-negative matrix factorization* (NMF) [59], which is applicable to neural network layers with non-negative activation, e.g. by using the popular *rectified-linear* activation function (section 2.2.2). Let $\boldsymbol{A} \in \mathbb{R}_{+}^{P \times M}$ be an intermediate representation produced by a convolutional neural network at one of its deep layers. NMF decomposes this

(a) Pyramids, $K = 4$



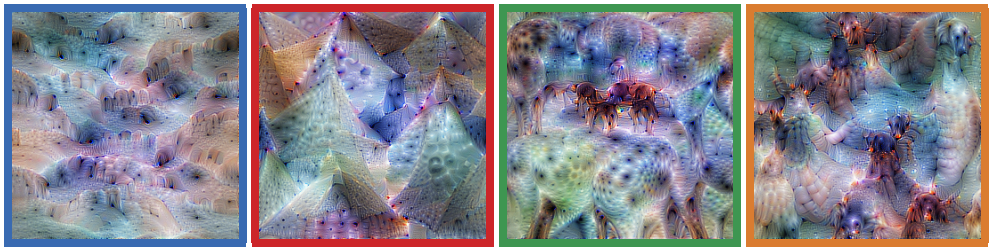
(b) Taj Mahal, $K = 3$

Figure 1.2 – *What in this picture is the same as in the other pictures?* Non-negative matrix factorization allows us to see how a deep CNN trained for image classification would answer this question. Here shown is an example of VGG-19 trained on ImageNet classification, and then applied to subsets from the iCoseg dataset. (a) Pyramids, animals and people correspond across images. (b) Monument parts match with each other.

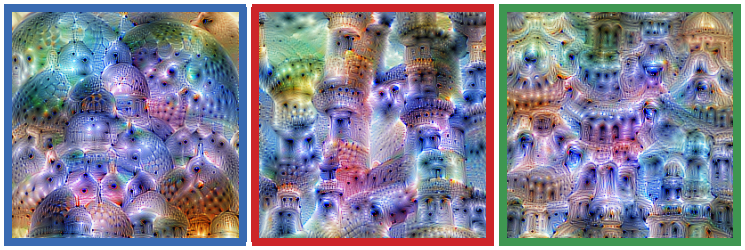
representation into two parts:

$$\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{V} \tag{1.1}$$

where $\boldsymbol{U} \in \mathbb{R}_+^{P\times K}$, $\boldsymbol{V} \in \mathbb{R}_+^{K\times M}$. When K is the smallest integer which maintains $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}$, it is called the *non-negative rank* of $\boldsymbol{A}$.



(a) Pyramids, $K = 4$



(b) Taj Mahal, $K = 3$

Figure 1.3 – Gradient ascent visualization of the features obtained through non-negative matrix factorization applied to VGG-19 activation for the *Pyramids* and *Taj Mahal* subsets from iCoseg. These visualizations of the rows of $\boldsymbol{V}$ correspond to the heatmap visualizations of the columns of $\boldsymbol{U}$ shown in Figure 1.2 using the same color encoding, i.e., the blue framed visualizations above correspond to the blue heatmaps in each corresponding row.

In Chapter 2 introduce interesting properties of NMF as well other factorization methods. In the same chapter we review deep neural networks, and convolutional networks in particular, with a focus on the challenges posed by generalization and interpretation.

In Chapter 3 we derive an upper bound over the amount of memorization in a network layer, expressed in terms of the non-negative rank of its activation matrices. We then explicitly apply compression to the intermediate representations learned by deep neural networks, which allows us to compare neural networks and determine which one is likely to generalize better - without the use of a validation set.

We show empirically that NMF is more effective than other matrix factorization techniques at distinguishing between memorization and generalization. Our method is also effective during training, which we demonstrate by performing *early stopping*, before overfitting commences.

Matrix factorization methods have been used for exploratory data analysis for decades. As a dimensionality reduction method, matrix factorization retrieves a compressed representation of the data, with fewer redundancies and correlations. When the dimensionality is sufficiently reduced, qualitative interpretation by human beings is made possible.

We show that this decomposition distills much of the semantics learned by a deep convolutional neural network. We roughly characterize $\boldsymbol{U}$ as "*where*" and $\boldsymbol{V}$ "*what*".

In Chapter 4, we use the $\boldsymbol{U}$ matrix of the NMF decomposition to produce heatmaps as in Figure 1.2, which offer an interpretable view into what information the network considers relevant and what information it discards. We find that for object recognition, the network learns to recognize fine-grained object *parts*, with invariance to complex transformations and noise. For instance, for the task of detecting *cars* we see that color is not a discriminative feature, whereas the presence or absence of *wheels* is.

We quantitatively demonstrate the rich semantics encoded in the matrix factors by applying them to real-world tasks such as co-localization and co-segmentation of a common object across a set of images. We obtain state-of-the-art results, with performance comparable or surpassing even hand-crafted domain-specific methods.

Each heatmap produced by the matrix $\boldsymbol{U}$ corresponds to a network feature which is stored in matrix $\boldsymbol{V}$. For instance, using feature visualization techniques can visualize $\boldsymbol{V}$ as shown in Figure 1.3. These visualization are informed of the original images shown in Figure 1.2 only through the features in $\boldsymbol{V}$.

In Chapter 5 we use the matrix $\boldsymbol{V}$ to describe an image globally as a *bag-of-concepts*. We evaluate this representation by performing *image search*. Specifically, we derive global image descriptors using $\boldsymbol{V}$ and show they can be used to retrieve images containing similar objects, with all the invariances learned by the network. For example, given an image of a *dog*, we search through a dataset and rank highly other images that portray *dogs*, regardless of scale, rotation and even *dog breed*.

By combining information from both $\boldsymbol{U}$ and $\boldsymbol{V}$ we perform *instance-based* search, i.e., given an image of a building, we rank highly only images of that particular building, and not images of other similar looking buildings. We again find that this approach outperforms comparable state-of-the-art methods.

In Chapter 6 we conclude the thesis with a summary of our contributions and directions for future work.

2 Related Work

2.1 Introduction

The methods proposed in this thesis concern various aspects of neural networks in general, and convolutional neural networks in particular. The first half of Section 2.2 therefore provides an introduction to neural networks, focusing on the necessity for multiple layers in order to achieve good performance on complex prediction tasks, and describing common design choices in modern networks.

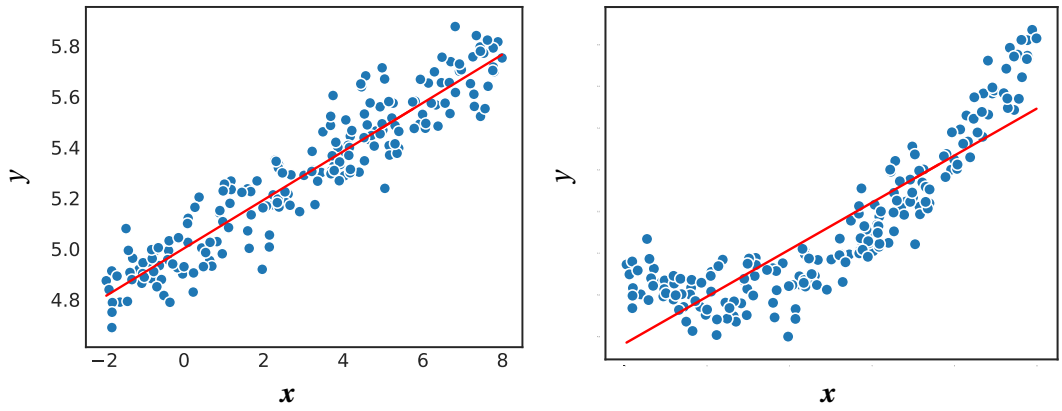
Section 2.2 then proceeds with an introduction of the two main themes of our work. We discuss empirical methods to detect *overfitting*, its impact on generalization, and theoretical work providing upper bounds on the *generalization error*. We then introduce the challenge of *network interpretability*, reviewing existing approaches with a focus on convolutional neural networks.

Our tool of choice for both detecting memorization and opening an interpretable window into convolutional neural networks, is *non-negative matrix factorization*. We introduce this method as well as other matrix factorization methods in Section 2.3.

2.2 Convolutional neural networks

2.2.1 From single neurons to deep convolutional neural networks

The class of computational models referred to as *artificial neural networks* (NNs) contains many diverse models. A common characteristic to all NNs, which they share with their biological namesakes, is the composition of simple computational units, the so-called neurons, into a larger collective, capable of computing more complex functions.



(a) (b)

Figure 2.1 – A single-neuron neural network with identity activation and mean squared error loss is equivalent to linear regression. Model predictions lie on the red line. (a) The data follows a linear distribution and the linear regression gives accurate predictions. (b) The data follows a non-linear distribution and linear-regression is too limited to give accurate predictions.

Linear-prediction: Single neuron and single layer

At its simplest, a single neuron is a function $\Lambda : \mathbb{R}^M \to \mathbb{R}$:

$$\Lambda(\boldsymbol{x}) = f(\boldsymbol{x}^\top \boldsymbol{w} + b) \tag{2.1}$$

i.e., a weighted-sum of the input vector $\boldsymbol{x}$ with weights $\boldsymbol{w}$, both in $\mathbb{R}^M$, followed by the addition of a bias term $b \in \mathbb{R}$. Finally the *activation function* f is applied to form the neuron output, called simply the *activation*.

A single-neuron NN can already retrieve common statistical models. We denote the output prediction of a NN as $\hat{y} = \Lambda(\boldsymbol{x})$. Letting $\mathcal{D} = \{\boldsymbol{x}_i, y_i | 1 \leq i \leq N\}$ be a training dataset of input-output samples, the weights $\boldsymbol{w}$ and the bias term b are trained to minimize the empirical loss:

$$\ell_{\mathcal{D}} = \frac{1}{N} \sum_{\boldsymbol{x}, y \in \mathcal{D}} \ell(\hat{y}, y) \tag{2.2}$$

$$\boldsymbol{w}^*, b^* = \underset{\boldsymbol{w}, b}{\arg\min}\, \ell_{\mathcal{D}} \tag{2.3}$$

where ℓ is the *loss* function, measuring the prediction error.

Different settings of activation and loss functions define different models. For instance, as shown in Figure 2.1, when $y \in \mathbb{R}$, *linear regression* can be cast as a single-neuron NN with

identity activation and squared-error loss:

$$f(a) = a \tag{2.4}$$
$$\ell(\hat{y}, y) = \|\hat{y} - y\|_2^2 \quad \rightarrow (\hat{y} - y)^2 \text{ if } y \in \mathbb{R}$$

Similarly, for binary classification with $y \in \{0, 1\}$, as in Figure 2.2, setting the activation function to the *logistic function* and loss to *binary cross-entropy* results in a single-neuron NN which performs logistic regression:

$$f(a) = \frac{1}{1 + e^{-a}} \tag{2.5}$$
$$\ell(\hat{y}, y) = -y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})$$

An example of multiple neurons composed together arises with the generalization of logistic regression to the case where $y \in \{1, \cdots, T\}$ and we wish to classify $\boldsymbol{x}$ as belonging to exactly one of T classes. This is accomplished by considering T neurons jointly, where we define the jth neuron output as the score or probability of belonging to the jth class. In this case, the weight vectors of the individual neurons are stacked into a single weight matrix, $\boldsymbol{W} \in \mathbb{R}^{M \times T}$, and similarly all bias terms to a vector, $\boldsymbol{b} \in \mathbb{R}^T$. Computing the NN output prediction proceeds as before:

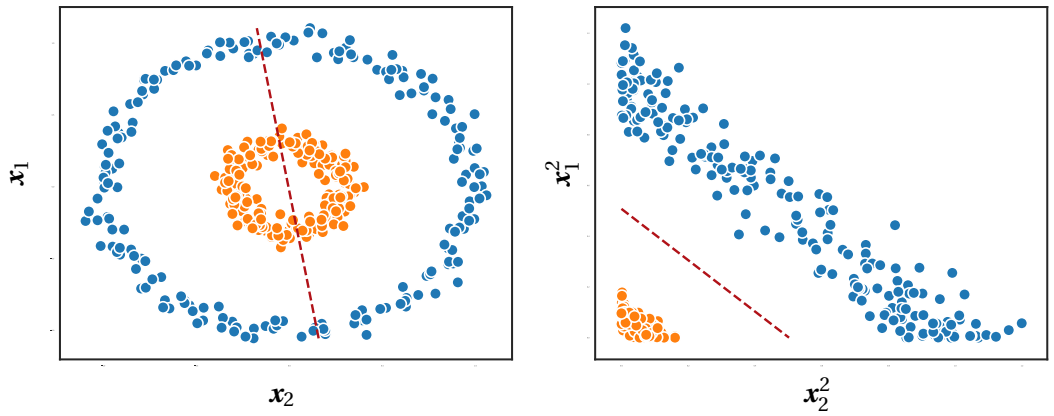
$$\hat{\boldsymbol{y}} = \Lambda(\boldsymbol{x}) = f(\boldsymbol{x}^\top \boldsymbol{W} + \boldsymbol{b}) \tag{2.6}$$

A suitable activation function in this case is the *softmax* function $f(\boldsymbol{x}) = \frac{e^x}{\|e^x\|_1}$, which normalizes its input to form a probability distribution. The cross-entropy loss in this case reduces to the negative log-likelihood of the correct class $\ell(\hat{\boldsymbol{y}}, y) = -\log \hat{\boldsymbol{y}}_y$. This model is referred to as softmax regression.

Neurons combined as in Eq. (2.6) are said to form a *layer* of width T. A layer parameterized this way is called "fully-connected", since every input affects every output. A fully-connected layer with M inputs and T outputs has a total $(M + 1) \cdot T$ trainable parameters.

An important property of the single-layer models described thus far is that they make *linear* predictions. This can be seen in Figures 2.1 and 2.2. Linear regression explicitly parameterizes a line equation on which its predictions lie, and both logistic and softmax regression are characterized by linear decision boundaries.

For classification, the linearity of decision boundaries does not pose a limitation if, indeed, the vectors associated with a label a, i.e., $\boldsymbol{X}_a = \{\boldsymbol{x}_i | y_i = a\}$, are *linearly separable* from the vectors $\boldsymbol{X}_b$, as in Figure 2.2b. In this case, logistic regression as described above is sufficient to fully



(a) (b)

Figure 2.2 – A single-neuron neural network with logistic activation and binary cross-entropy loss is equivalent to logistic regression. The decision boundary at $\hat{y} = 0.5$ is shown in red. (a) Logistic regression cannot separate clusters which are not linearly-separable. (b) An appropriate transformation can extract features that result in linear-separability.

minimize the loss.

However, if $\boldsymbol{X}_a$ is *not* linearly-separable from $\boldsymbol{X}_b$, as in Figure 2.2a, inherent model error is introduced that cannot be overcome by any setting of the parameters $\boldsymbol{W}$ and $\boldsymbol{b}$.

Feature extraction: multiple layers

To overcome this issue, the input vectors $\boldsymbol{x}$ must first be transformed, by an additional function Λ_h, into having linearly-separable structure. The transformed output, called *features*, can then be fed to the single-layer classifier:

$$\hat{\boldsymbol{y}} = \Lambda(\Lambda_h(\boldsymbol{x})) \tag{2.7}$$

In the case shown in Figure 2.2a, although the data is given as two dimensional vectors $\boldsymbol{x} =< x_1, x_2 >$, the cluster identity is determined by the radius length $x_1^2 + x_2^2$. Setting $\Lambda_h(\boldsymbol{x}) =< x_1^2, x_2^2 >$ is therefore an adequate transformation, which yields the linearly-separable clusters shown in Figure 2.2b.

The question remains, however, how to find appropriate features without requiring case-by-case analysis. Such analysis typically involves considerable investment and is limited by the level of domain-specific expertise available to the feature designer.

One solution is to augment the input with higher-order interactions, e.g., the squared inputs used in the above example. It is unclear, however, how complex these interactions must be in order to achieve linear separability. Furthermore, adding all interaction terms of a certain order increases the input size combinatorially, which quickly makes the weight matrix $\boldsymbol{W}$ large. Over-parameterized layers have higher space and time complexity, and are exposed to the danger of *overfitting*, discussed in section 2.2.3.

A different approach is to model the function Λ_h similarly to the function Λ:

$$\Lambda_h(\boldsymbol{x}) = g(\boldsymbol{W}_h^\top \boldsymbol{x} + \boldsymbol{b}_h) \tag{2.8}$$

where g is an activation function and $W_h \in \mathbb{R}^{M\times P}$. We accordingly redefine $\boldsymbol{W} \in \mathbb{R}^{P\times T}$.

The model prediction now becomes a composition:

$$\hat{\boldsymbol{y}} = \Lambda(\Lambda_h(\boldsymbol{x})) \tag{2.9}$$

and the parameters of Λ_h and Λ are both subject to training. Analyzing the features extracted by Λ_h is an interesting challenge, as discussed in Section 2.2.4.

In the resulting *two-layer* model, Λ is called the *output layer* and Λ_h is referred to as a *hidden layer*, since its P-dimensional output activation is a latent code without any a-priori assigned meaning. The term *input layer* is sometimes used to refer to the input $\boldsymbol{x}$ itself. This layered architecture is called *feed-forward*, since obtaining the prediction involves feeding the output of each layer forward to the next layer.

The layer Λ_h has limited expressive power, since it consists of only a single linear transformation followed by an element-wise activation function. Achieving linear-separability may require more complex transformations. In this case, complexity could be increased by introducing additional layers to the model, increasing its *depth*.

We thus arrive to the general L-layer feed-forward NN model, which is the underlying paradigm for most modern NN architectures today:

$$\begin{aligned} \boldsymbol{a}_0 &= \boldsymbol{x} \\ \boldsymbol{a}_i &= \Lambda_i(\boldsymbol{a}_{i-1}), \quad 1 \le i \le L \\ \hat{\boldsymbol{y}} &= \boldsymbol{a}_L \end{aligned} \tag{2.10}$$

The number of hidden layers, $L-1$, their widths, and the activation functions used are hyper-parameters defining the NN architecture, and are set by the network designer.

The vector $\boldsymbol{a}_i$ is the activation vector at layer i, also referred to as the *features* extracted by the first i layers. The activation of the final layer is the output prediction $\hat{\boldsymbol{y}}$.

The intuition behind *deep learning* suggests that the deeper the layer i, the more abstract and semantically meaningful are its features [12].

Convolution

Computer vision deals with image data for tasks such as object recognition and segmentation. Using fully-connected NNs to solve these tasks quickly runs into a scalability issue: image data can have hundreds-of-thousands or millions of dimensions, even at moderate image resolutions.

To resolve this, the *convolutional* neural network (CNN) was introduced [58]. In this network the basic layer computation is changed to have two properties: local-connectivity and translation-invariance. These properties drastically reduce the number of network parameters, thereby reducing computational load and the danger of overfitting.

Local-connectivity means that by applying small filters to image patches, only interactions between near-by pixels are considered. This is based on the assumption that, for instance, a pixel near the top-left corner of the image and a pixel near the bottom-right corner, are unlikely to be related in a consistent manner which can be useful for making predictions.

Translation-invariance means we can use the same local filter in all spatial positions. This is based on the assumption that image statistics distribute identically across the spatial dimensions of the image.

With the properties combined, the result is that a CNN layer performs a 2D spatial-convolution (or cross-correlation) between the input image and a set of learned filters. Note that the same local-connectivity and translation-invariance assumptions can be applied to any directional data, e.g., time-series or text data, in which case convolution and filters are 1D.

Formally, an image is represented as a tensor $\mathbf{I} \in \mathbb{R}^{C_\mathbf{I} \times H_\mathbf{I} \times W_\mathbf{I}}$, where $H_\mathbf{I}$ and $W_\mathbf{I}$ are the height and width of the image respectively, and $C_\mathbf{I}$ is the number of channels, e.g., $C_\mathbf{I} = 3$ for RGB color images.

The weight tensor of a convolutional layer $\mathbf{W} \in \mathbb{R}^{C_{\text{out}} \times C_{\text{in}} \times H_\mathbf{W} \times W_\mathbf{W}}$ is viewed as a set of C_{out} filters, each of spatial dimensions $H_\mathbf{W} \times W_\mathbf{W}$ and as many channels as the layer input. The bias vector $\boldsymbol{b} \in \mathbb{R}^{C_{\text{out}}}$ has a value for every output channel, which is added at all spatial positions.

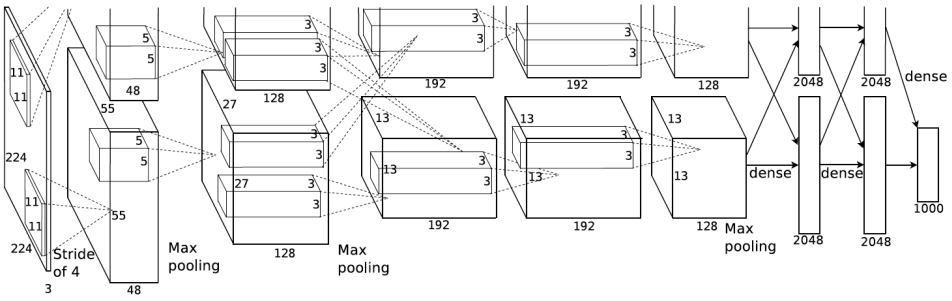


Figure 2.3 – A schematic of AlexNet. Figure taken from Krizhevsky et al. [55].

A convolutional layer operates in the image's 2D space and computes:

$$\begin{aligned} &\mathbf{A}_i \in \mathbb{R}^{C_{\text{out}} \times H_{\mathbf{A}_i} \times W_{\mathbf{A}_i}} \\ &\mathbf{A}_i = \Lambda_i(\mathbf{A}_{i-1}) \text{ such that } [\Lambda_i(\mathbf{A}_{i-1})]_c = f(\mathbf{A}_{i-1} * \mathbf{W}_c + \boldsymbol{b}) \end{aligned} \tag{2.11}$$

where $*$ denotes convolution, and $\mathbf{A}_{i-1}$ is the layer input, e.g. $\mathbf{A}_{i-1} = \mathbf{I}$, and thus $C_{\text{in}} = C_{\mathbf{A}_{i-1}}$.

Each of the C_{out} channels of the resulting activation tensor, $\Lambda(\mathbf{X})$, is also called a *feature map*, since it is a 2D activation map where each value represents a feature response with respect to an image patch.

The output dimensions $H_{\mathbf{A}_i} \times W_{\mathbf{A}_i}$, being a function of $H_{\mathbf{I}}$ and $W_{\mathbf{I}}$, vary from image to image. In general, convolution produces an output with decreased spatial dimension. To account for this, padding is added at each layer to give an output with the same spatial dimensionality as the input.

Commonly, spatial dimensions are reduced with explicit down-sampling, accompanied by an increase in the channel dimension. This further reduces the memory and computation requirements. Down-sampling is accomplished by setting the stride of convolutions or *pooling* operations to be greater than one.

A common form of pooling is *max-pooling*. Defined by a window of size $H_p \times W_p$, the pooling window slides over the image akin to the convolution operation, and returns at every position the maximal value within the window. When applied to an activation tensor with C channels, max-pooling is applied to each channel separately, i.e., given input $\mathbf{A} \in \mathbb{R}^{C \times H_p \times W_p}$, max pooling returns a $C - dimensional$ vector, having pooled the spatial dimensions.

Arguably the first successfully trained large-scale CNN is AlexNet [55]. It consist of several convolutional layers, followed by several fully-connected layers. A schematic of AlexNet is shown in Figure 2.3.

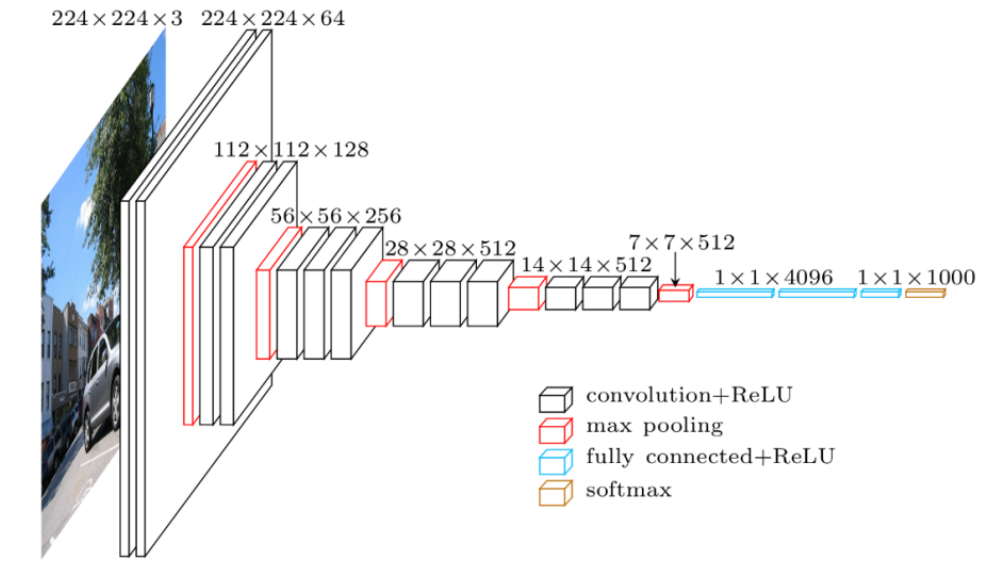


Figure 2.4 – A schematic of VGG16. VGG networks follow the design of AlexNet, but are deeper and exclusively use small filters. Figure taken from blog.heuritech.com.

Trained on ImageNet [80], AlexNet classifies images as belonging to one of 1K classes. With a total of 60M parameters, this network achieved a top-1 accuracy of 62.5% and top-5 accuracy of 83%.

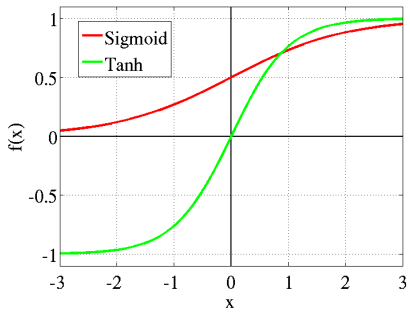
The networks VGG-16 and VGG-19 [87] closely follow the architecture of AlexNet. However, they exclusively use small 3×3 convolutional filters (compared to 11×11 in the first layer of AlexNet), and have increased depth. VGG-16 improved CNN performance on ImageNet to classification to top-1 accuracy of 74.4% and top-5 accuracy of 91.9%. See Figure 2.4 for a schematic.

2.2.2 Training and gradient flow

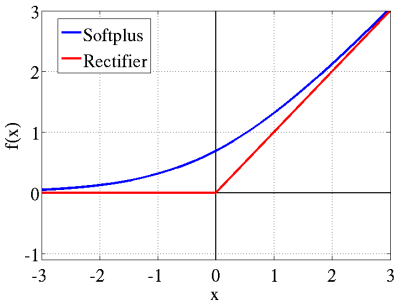
Gradient optimization

Minimizing the objective function of the logistic regression model of Eq. (2.4) can be done analytically: computing the gradient of the loss with respect to $\boldsymbol{w}$ and setting it to zero gives the well-known *normal equation*, which has a closed-form solution yielding the optimal $\boldsymbol{w}$.

Logistic regression, in Eq. (2.5), on the other hand, does not lend itself to a closed-form solution. Instead, its parameters are iteratively updated with *gradient descent* (GD). Starting from a random parameter initialization, at every iteration the gradient $\nabla_{\boldsymbol{w}} \ell$ is computed, and



(a) Sigmoid activation functions



(b) Rectified-linear activation function

Figure 2.5 – The activation function plays a crucial role in the convergence properties of a deep NN. (a) Saturating activation functions have small gradients far from the origin, which slows down learning as NN weights grow in magnitude. (b) The rectified-linear function [37], on the other hand, has a gradient of 1 for all positive activations. The non-negativity of the rectified-linear function enables us to use NMF in our analysis of CNN activations. Figure taken from Glorot et al. [37].

the weights are updated as:

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \alpha \nabla_{\boldsymbol{w}} \tag{2.12}$$

where α is a hyper-parameter controlling the *step size*.

When the step size is not too large, GD is guaranteed to find the globally optimal value of $\boldsymbol{w}$, because the single-layer logistic regression model has a *convex* objective function.

A NN or CNN with many layers, however, defines a highly non-convex function. Minimizing such a function with GD could theoretically "get stuck" in a sub-optimal local minimum.

With the availability of large datasets for training, *stochastic* gradient descent (SGD) has been proposed, which performs a gradient update based on a stochastically sampled subset of the data, called a *batch*. The introduction of sampling noise could account for the good performance of (stochastic) gradient descent methods in practice, since the noise allows the network to escape from narrow local minima [51].

Augmentations of SGD have also been proposed, such as SGD with momentum [74], Adagrad [29] and ADAM [52]. These methods all maintain a per-parameter learning rate, resulting in faster convergence than standard SGD.

One of the main challenges of optimization with SGD variants is the problem of *vanishing and exploding gradients*. Specifically, the gradient is computed by back-propagating the error

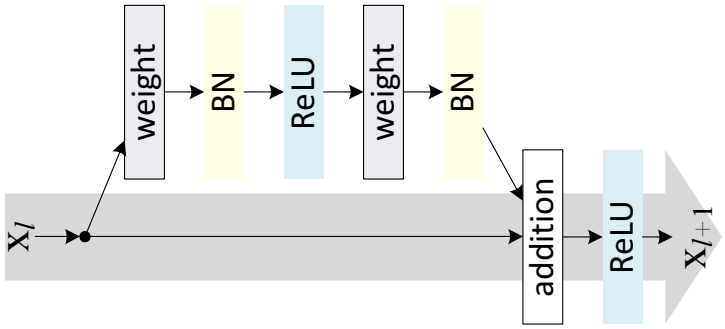


Figure 2.6 – Residual connections form the basis of ResNets. In this diagram *weight* refers to a linear transformation, BN to batch-normalization,and ReLU is the rectified-linear function. Figure taken from He et al. [44].

[79] from the output layer, through deep layers, into early layers. Through multiplicative interactions along the way, the gradient signal for early layers can have a very small or very large magnitude, leading to slow convergence. The next two sections deal with proposed solutions to this phenomenon.

Activation functions and residual connections

The activation function produces the final layer output and can greatly affect gradient flow. For many years, sigmoidal ("S"-shaped) functions, such as the logistic function and the hyperbolic tangent were widely used, shown in Figure 2.5.

With the advent of deep networks, it was soon realized that sigmoid functions caused gradients to vanish. As network weights grew with every SGD update, activations would be pushed into the high- and low-end plateaus of the sigmoid activation function, where the gradient is very small.

A solution was proposed in the form of rectified-linear units (ReLU) [37]. This activation function, $f(x) = \max(x, 0)$, has a gradient that is either 0 or 1, as shown in Figure 2.5b. This allows gradients to flow freely (or not at all), and is now standard in most modern CNNs. ReLU additionally has the property of producing non-negative activations, which enables our analysis of CNN activations using non-negative matrix factorization.

While ReLUs can pass the gradient unaltered, a long chain of linear transformations is still sufficient to hurt network performance. A solution to this issue comes in the form of *residual connections* [44], which tweaks the standard feed-forward pipeline as follows:

$$\boldsymbol{a}_i = \boldsymbol{a}_{i-1} + \Lambda_i(\boldsymbol{a}_{i-1}), \quad 1 \leq i \leq L \tag{2.13}$$

In other words, the layer Λ_i computes a *residual* term, which, once added to the layer input, achieves the desired transformation. Figure 2.6 shows the basic residual block.

This simple change means that the gradient can flow into $\boldsymbol{a}_{i-1}$ through Λ_i as well as *directly*. The popular ResNet-50, which consists of 50 residual layers, further improves upon the standard feed-forward nets described above, achieving ImageNet classification top-1 accuracy of 79.26% and top-5 accuracy of 94.75%. In addition to residual connections, ResNets employ ReLU activations and *bach-normalization*, described below.

Initialization and batch-normalization

The random initialization at the start of training has been found to have a significant impact on convergence [64]. Effective initialization methods have been proposed to ensure that the gradient magnitude is maintained constant as it flows from layer to layer [36; 43], improving convergence. These methods, however, only affect SGD iterations near the start of training.

Batch-normalization [46], on the other hand, is a method that explicitly normalizes network activations according to the statistics of the current SGD batch, all throughout training. By doing so, the magnitude of the gradients is also controlled, and convergence is accelerated. This technique also is widely incorporated into most modern CNNs, and is applied between the convolution and ReLU activation.

2.2.3 Generalization and overfitting

The training set $\mathcal{D} = \{\boldsymbol{x}_i, y_i | 1 \leq i \leq N\} \sim \{X \times Y\}$ is a collection of N samples, each sampled from the *data distribution* $X \times Y$ over pairs $(\boldsymbol{x}, y)$. In the discussion so far we have referred to NN training as the task of minimizing the empirical loss, or *training* loss, evaluated using $\mathcal{D}$, as in Eq. (2.2).

The ultimate goal of learning, however, is to do well on new data, unseen during training. We want to minimize the *generalization loss*:

$$\ell_{X \times Y} = \mathbb{E}_{\boldsymbol{x}, y \sim X \times Y}[\ell(\Lambda(\boldsymbol{x}), y)] \tag{2.14}$$

The generalization loss is approximated by measuring the *test loss* using a test set $\mathcal{D}_{\text{test}}$, which contains additional samples from $X \times Y$ not used during training.

Large NNs have the capacity to *overfit* their training set, i.e., *memorize* individual input-to-output mappings, which does not promote good generalization to new data samples. In this way, the training loss can be minimized while the test loss remains high. This phenomenon is demonstrated in Figure 2.7.

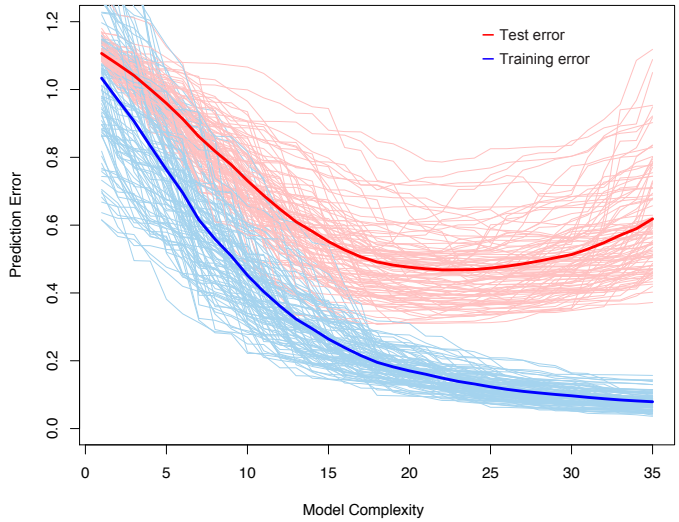


Figure 2.7 – Overfitting is characterized by low training error but high test error. This is due to a model having sufficient capacity to memorize individual input-to-output mappings, at the expense of converging onto a more general strategy. This figure is adapted from Friedman et al. [34], where 100 datasets were reconstructed using sparse coding. Shown in light blue, the training prediction error of individual trials steadily decreases with increased model complexity (e.g., dictionary size). However, when examining the test prediction error, shown in light red, there is an intermediate model complexity above which overfitting starts to occur. Average training and test errors are show as solid lines.

It has been observed empirically that modern CNNs have sufficient capacity to memorize even very large datasets with random labels [102], but when trained on real data they generalize well, a behavior which seemed to defy theoretical justification.

Early bounds on $\ell_{X\times Y}$ were derived based on properties such as the Vapnik–Chervonenkis (VC) dimension [92] and Rademacher complexity [7]. These properties indicate a model's ability to fit randomly labeled data, i.e., $\tilde{\mathcal{D}} = \{\boldsymbol{x}, \tilde{y} | \boldsymbol{x} \in \mathcal{D}\}$, where $\tilde{y}$ is a randomly (re-)sampled label. They can therefore serve as an estimate of model complexity, or more specifically the complexity of the decision boundaries it can produce. However, since these quantities depend on the number of network parameters, these bounds do not explain the good performance of NNs, which in modern networks can reach billions.

Generalization bounds were recently proposed which scale with the number of NN layers [8; 69], as a product of weight matrix norms, but modern NNs have shown *better* performance with increased depth, which therefore remains unexplained.

Other work has focused on properties of SGD itself, showing it is biased towards minima that

generalize well [88; 15]. These results relate to the *flatness of the local minimum* to which the network converges.

In a flat minimum, applying perturbations to a network's parameters or activations does not result in a dramatic change to its (training set) performance, and this is taken as an indicator of good generalization [45; 17; 51; 68; 61]. Robustness to noise was used also to derive generalization bounds [2].

However, any reversible transformation, such as simple scaling, can be used to arbitrarily manipulate the curvature of the local minimum, without affecting generalization [28]. For instance, analysis could be applied to layer Λ_i with perturbations of a certain magnitude, and its sensitivity could be measured. Then, dividing the weights of Λ_i by 10 and multiplying the weights of Λ_{i+1} by 10 does not change the output of a ReLU NN, and so does not affect its generalization. However, subjecting the re-scaled Λ_i to the same magnitude of perturbations as before will show it as being more sensitive. Care must therefore be taken to use perturbations of 'appropriate units' [61].

Related to the notion of flat local minima, *compression* has been proposed to play a key role for good generalization [2]. Our analysis in Chapter 3 pursues this direction explicitly, where we study the impact of compression, through matrix factorization, on the performance of NNs.

Notably, the *information bottleneck principle* (IB) [84] proposes that well-generalizing NNs have hidden layers which function as *minimal sufficient statistics* [86]. Formally, let the random variable X represent the NN input and Y its output. A hidden variable Z is a function of the input, and by the data-processing inequality [24]:

$$\mathcal{I}(X,Y) \geq \mathcal{I}(X,Z) \tag{2.15}$$

where $\mathcal{I}(\cdot,\cdot)$ is the *mutual information*.

In other words, the input X itself contains all the relevant information about the output Y. The hidden representation Z is considered optimal according to IB if it discards irrelevant information about X, retaining only the information relevant to the prediction of Y:

$$Z = \arg\max_{Z} \mathcal{I}(Y,Z) - \beta\mathcal{I}(X,Z) \tag{2.16}$$

where β controls the level of compression. Optimizing empirical approximations of the compression terms has shown to improve generalization in practice [11].

By virtue of using mutual information, IB provides a generalization bound that is invariant to reversible transformations, addressing the issue raised in perturbation analysis methods. This advantage carries over to our own analysis, and indeed, our dimensionality-reducing

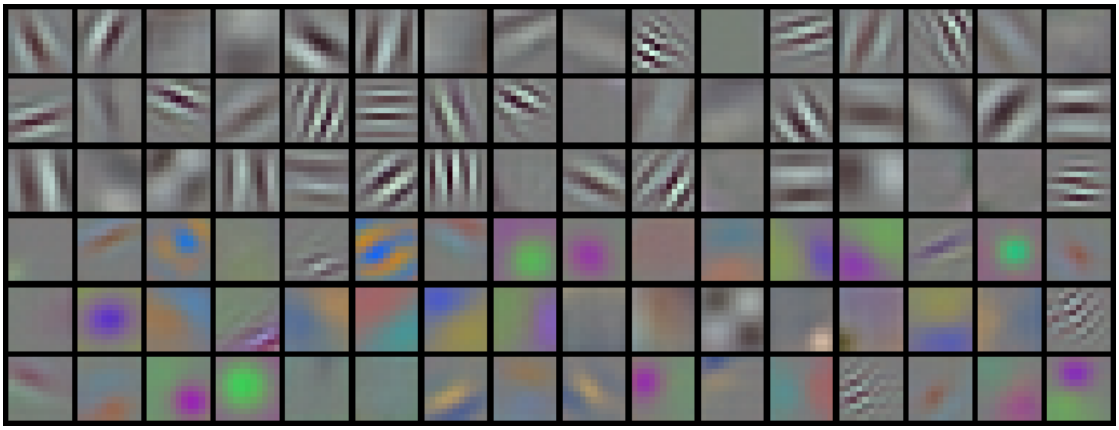


Figure 2.8 – First layer CNN filters can be directly visualized, often revealing Gabor-like filters and opponent-color edges. Depicted here are the first-layer filters of AlexNet [55].

perturbations are similarly invariant.

2.2.4 Network interpretability

In section 2.2.1 we introduced the concept of automatic feature extraction. Each layer of a neural network extracts more complex features, which eventually indicate whether a sample belongs to a particular output class or another.

A few things are known a-priori. The activation $\boldsymbol{a}_0$ is the input itself. The activation $\boldsymbol{a}_L$ represents scores or probabilities of an input belonging to each output class. The *feature space*, i.e., the space of activation vectors, of layer Λ_{L-1} should be such that clusters of activation vectors $\boldsymbol{a}_{L-1}$ belonging to different clusters are linearly separable.

Otherwise, not much is known about the intermediate layers. What features do the hidden-layer activations $\boldsymbol{a}_2$ or $\boldsymbol{a}_{L-2}$ extract? What information does the network use to solve a prediction task? The lack of clear answers to these questions has led to NNs being described as black-box models.

As NNs are incorporated into applications requiring validation and user trust, such as medical diagnosis [16] and autonomous driving [13], there is a growing need to answer these questions. The field of *network interpretability* has therefore received much attention recently.

Specifically for CNNs applied to image data, the task of interpretation is somewhat simplified. Various visualization techniques have been proposed that lend themselves to qualitative interpretation by humans.

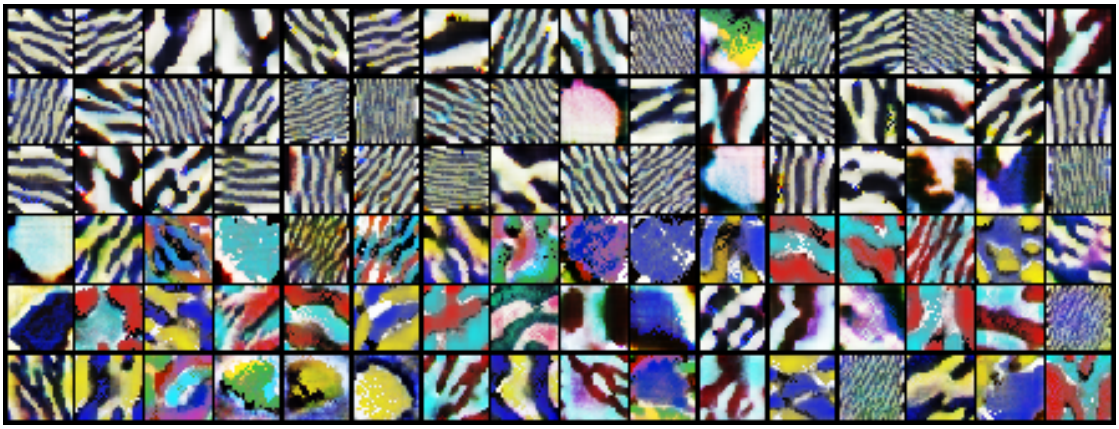


Figure 2.9 – Gradient ascent visualization of the AlexNet first-layer filters shown in Figure 2.8.

Feature visualization

First layer filters exist in pixel space and can be visualized. For instance, the first layer of AlexNet consists of 96 filters of size 11×11 and 3 color channels. Shown in Figure 2.8, these filters are reminiscent of the well-known result of Olshausen and Field [71], where sparse-coding applied to natural images retrieved filters suggestive of the receptive field of biological neurons in the V1 region of the human visual cortex.

The features extracted by deeper layers do not have a pixel representation, and so cannot be similarly visualized. However, it is possible to *generate* an input in pixel space, that once fed to the CNN results in a high filter response value for a filter of interest [31]. Visualizing the feature corresponding to the jth filter in the ith layer amounts to finding:

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x}} \| \left[\Lambda_i \circ \cdots \circ \Lambda_1(\boldsymbol{x})\right]_j \| \tag{2.17}$$

This optimization can be accomplished with *gradient ascent*, starting from a random $\boldsymbol{x}$ and iteratively updating it. For ReLU networks, since activation in not bounded from above, a norm constraint is added on $\boldsymbol{x}$. Many additional regularization tricks have been proposed to improve feature visualization [63; 70].

To get a sense of this type of visualization, we use the method of Olah et al. [70] to generate visualization for the first layer filters shown in Figure 2.8 and qualitatively compare. The corresponding gradient ascent visualizations are shown in Figure 2.9. As can be seen, these visualizations are able to roughly capture the color-selectivity and frequency of the filters, but introduce a significant amount of random variation.

When applied to a deep layer, feature visualization can reveal a degree of semantics. In Figure

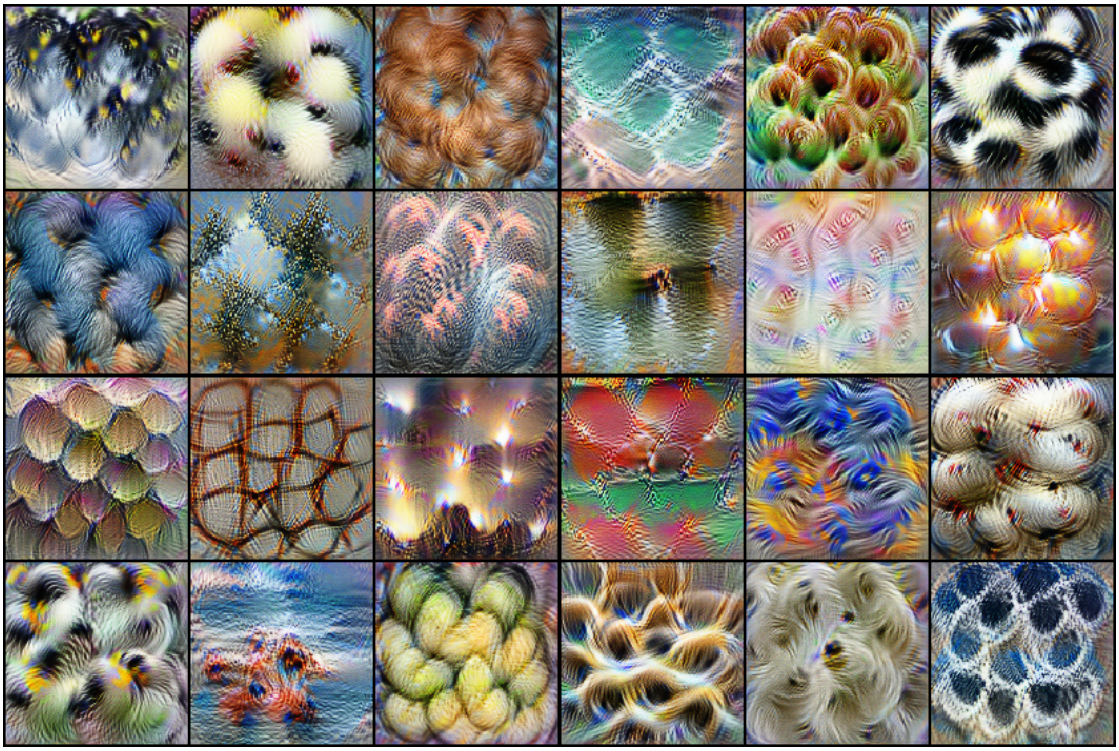


Figure 2.10 – Gradient ascent allows us to visualize features in deep layers. Here we visualized a subset of filters from the fifth and final convolutional layer of AlexNet. As opposed to the rudimentary features of the first layer (Figure 2.9), these features show more variation and can reveal a degree of semantics. For instance, the filter visualized on the bottom row, second from the right, seems to capture a feature related to *dogs*.

2.10 we apply feature visualization to the fifth and final convolutional layer of AlexNet. Some of the resulting images are suggestive of various objects and landscapes[1].

Note that the gradient ascent technique is not limited to visualizing single filters, but rather any weighted combination of their responses. In general, whether the axis aligned directions in feature space (i.e., single filters) are more meaningful that off-axis directions is an open question. However, it is not clear a-priori which combination of filters is meaningful. As we will show in the sequel, such directions of interest can be found using NMF.

Saliency maps

A different class of methods, exploits the fact that convolutional layers preserve the 2D layout of the image. To undo the down-sampling induced by the CNN architecture, feature maps of

[1] If it is not immediately clear, we recommend squinting.

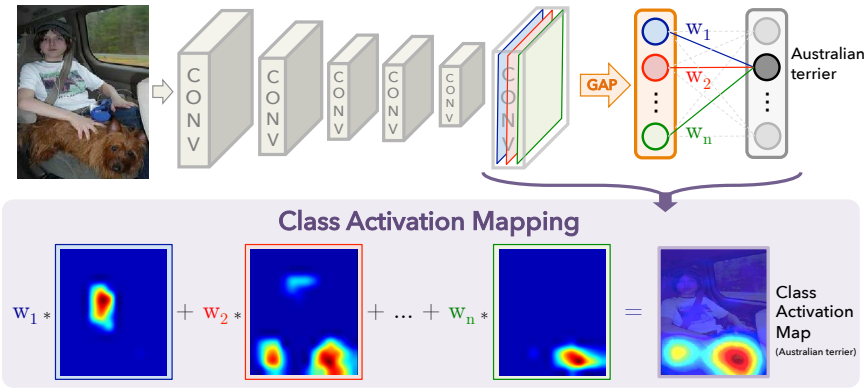


Figure 2.11 – CAM produces a saliency map with respect to a specific output class. The final weight vector entering the output unit corresponding to the class of interest is used to average CNN feature maps. Figure taken from [103].

a deep layer are up-sampled back to the original image resolution. When overlaid with the input image, the up-sampled feature map acts as a *heatmap*, spatially highlighting the image region which the corresponding filter responds to.

This is the basis of "network dissection" [10]. Having acquired images with pixel-wise ground truth annotation, indicating object class, object-part class (e.g., *leg*), texture and color, the authors proceed to test each individual feature map for significant overlap with any ground truth concept.

This method suffers from two main limitations. First, the availability of ground truth is essential to their analysis, and limits the concepts which can be detected. Second, it does not consider filter combinations, though these might correspond to concepts which otherwise would be missed.

One example of an approach which creates a single heatmap by performing a weighted combination of feature maps is *class-activation maps* (CAM) [103]. CAM requires a specific CNN architecture, and therefore cannot be applied to AlexNet and VGG.

The CAM pipeline is shown in Figure 2.11. It requires a CNN classifier that uses *global average pooling* (GAP) to transition to the final fully-connected layer. That is, the feature maps of the last convolutional layer are averaged spatially to yield a vector of fixed-dimension. CAM then produces class-aware saliency maps, by averaging the feature maps (*before* pooling) using the weights of the fully connected layer which correspond to the output class of interest.

CAM can be seen as a special case of using the gradient with respect to a specific output unit to derive weights for combining features maps. This observation has led to the more

general Grad-CAM method [83], which can be applied to any CNN architecture. Grad-CAM is still, however, limited to producing maps with respect to the set of output classes used for training the network. Layer-wise Relevance Propagation [56] produces heatmaps similarly to Grad-CAM, but propagates a *relevance score* instead of the gradient.

The heatmaps we derive with NMF, however, are not associated with an output unit or output class. Instead, NMF heat maps capture common activation patterns, which allows us to create saliency maps which localize objects never seen before by the CNN, and for which there is no designated output unit.

2.3 Matrix factorization

Matrix factorization techniques describe each data point in a matrix $\boldsymbol{A} \in \mathbb{R}^{N \times M}$ as a combination of K basis vectors:

$$\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{V} \tag{2.18}$$

where $\boldsymbol{U} \in \mathbb{R}^{N \times K}$ and $\boldsymbol{V} \in \mathbb{R}^{K \times M}$. In this paper we consider factorizations minimizing the Frobenius or ℓ_2 norm of the reconstruction error, i.e., $\|\boldsymbol{A} - \boldsymbol{U}\boldsymbol{V}\|_F$.

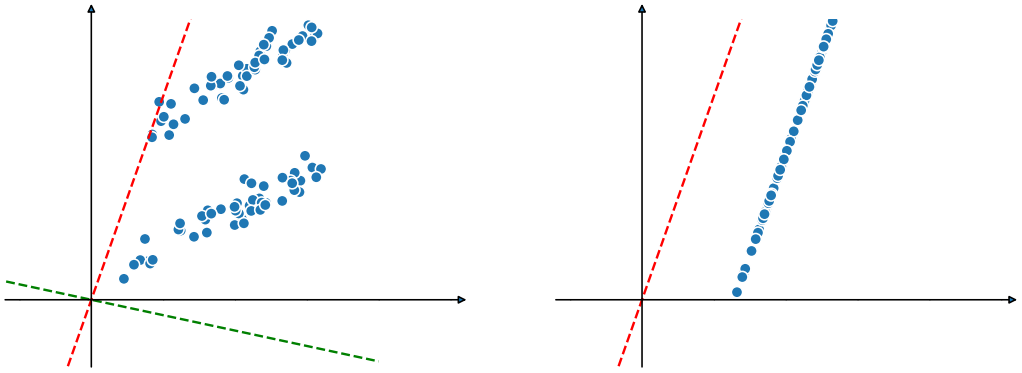
Factorization can give zero reconstruction error as long as K is greater or equal to the *intrinsic dimension* of the data, i.e., the dimension of the data manifold encoded by the matrix $\boldsymbol{A}$. When K is smaller than the intrinsic dimension, lossy compression occurs.

In addition to limiting the number of basis vectors K, different factorization methods impose additional constraints on what these vectors can be and how they can be combined. We consider several factorization methods, introduced below.

2.3.1 Principal component analysis (PCA)

A classical method for dimensionality reduction is principal component analysis (PCA) [48]. Projecting the data from the low-dimension space back to the original space give an approximation that is constrained in its rank. As a matrix factorization method, then, PCA finds an approximation that is optimal (in the ℓ^2 sense):

$$\begin{aligned} \mathrm{PCA}(\boldsymbol{A}, K) = \underset{\hat{\boldsymbol{A}}_K}{\arg\min} \quad & \|\boldsymbol{A} - \hat{\boldsymbol{A}}_K\|_F^2, \\ \text{subject to} \quad & \hat{\boldsymbol{A}}_K = \boldsymbol{A}\boldsymbol{V}\boldsymbol{V}^\top, \\ & \boldsymbol{V}^\top\boldsymbol{V} = I_K, \end{aligned} \tag{2.19}$$



(a) $K = 2$ (b) $K = 1$

Figure 2.12 – Principal component analysis in 2D. (a) Setting $K = 2$ naturally returns an orthonormal basis that spans $\mathbb{R}^2$, and so the 2D data is perfectly reconstructed. (b) Setting $K = 1$ projects the data onto a 1-dimensional subspace, leading to lossy compression.

where $V_K \in \mathbb{R}^{M \times K}$ is an orthonormal basis.

PCA has an *incremental* property, meaning the K basis vectors of a K-rank approximation are also the first K basis vectors of a $(K+1)$-rank approximation. The smallest K that admits an exact PCA approximation $\boldsymbol{A} = \hat{\boldsymbol{A}}_K$ is the *rank* of $\boldsymbol{A}$, $\text{rank}(\boldsymbol{A})$.

An example of PCA in 2D is shown in Figure 2.12.

2.3.2 k-means

Though normally viewed through the lens of clustering, k-means can be seen as a factorization method, optimizing the following objective:

$$\begin{aligned} \text{k-means}(\boldsymbol{A}, K) = \underset{\hat{\boldsymbol{A}}_K}{\arg\min} \quad & \|\boldsymbol{A} - \hat{\boldsymbol{A}}_K\|_F^2, \\ \text{subject to} \quad & \hat{\boldsymbol{A}}_K = \boldsymbol{U}\boldsymbol{V}^\top, \\ & [\boldsymbol{U}]_i \in \{\boldsymbol{e}_1, \cdots, \boldsymbol{e}_K\}, \end{aligned} \tag{2.20}$$

where $\boldsymbol{U} \in \mathbb{R}^{N \times K}$, $H \in \mathbb{R}^{K \times M}$, and $\boldsymbol{e}_j$ is the canonical vector of dimension j. As such, a row in $\boldsymbol{A}$ in approximated by exactly one row of $\boldsymbol{V}$.

As can be seen, although the constrains are different, the basic objective is the same as PCA [27]. An example of k-means in 2D is shown in Figure 2.14.

From a probabilistic point of view, k-means is equivalent to performing maximum-likelihood

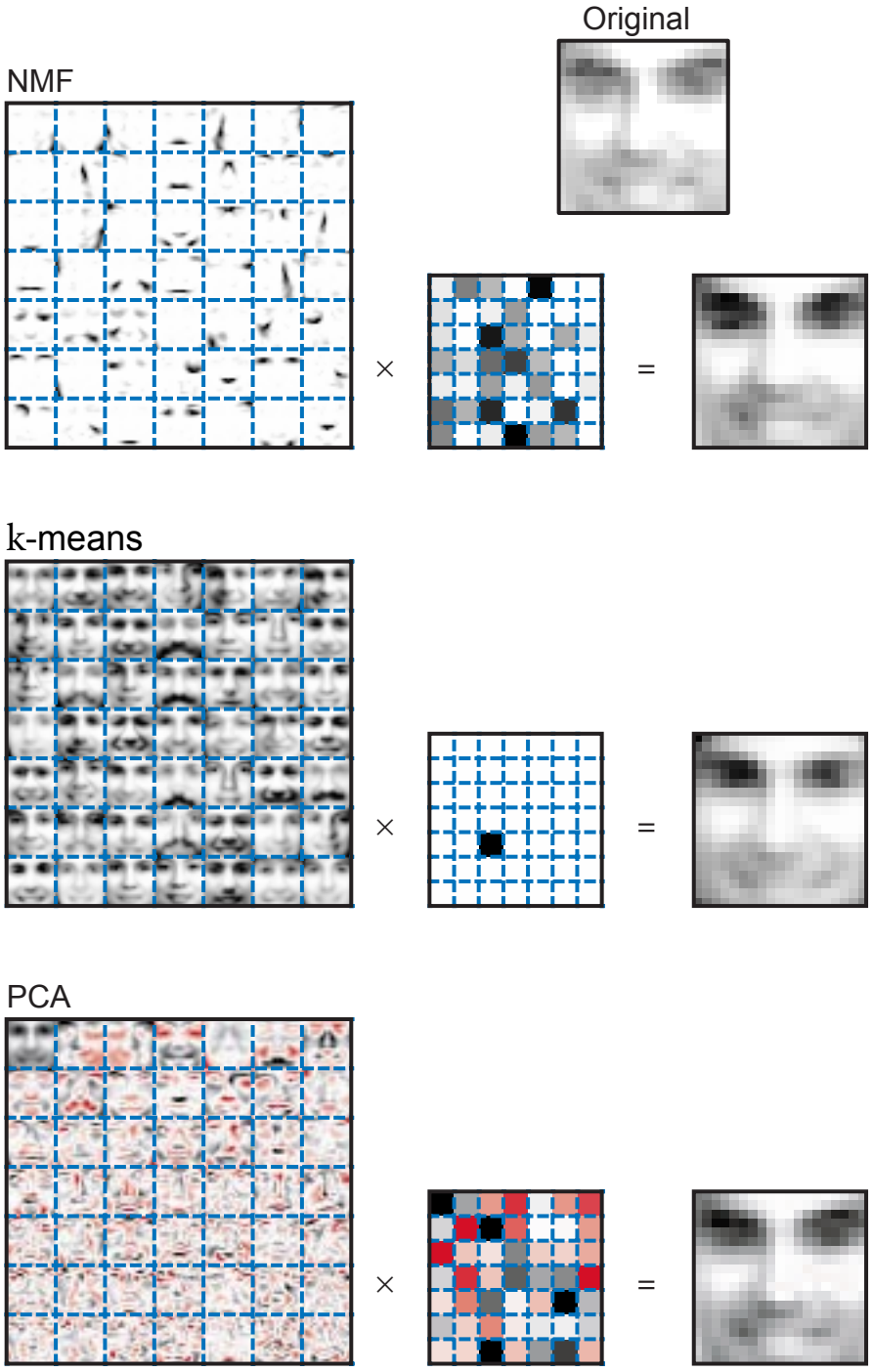
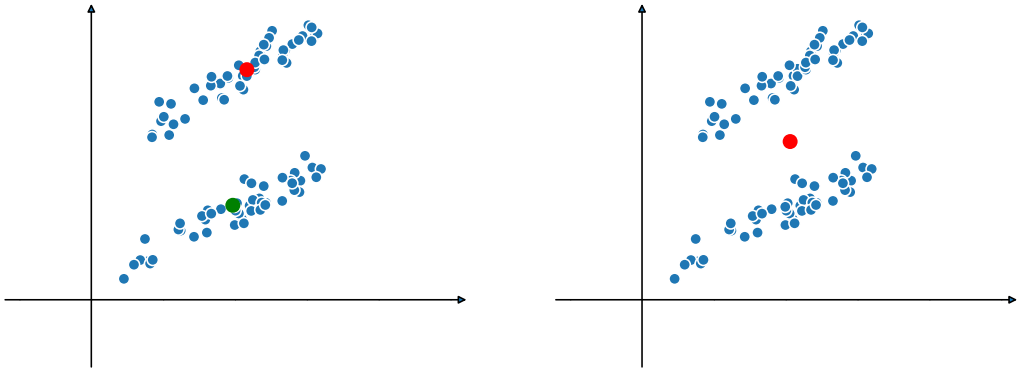


Figure 2.13 – Non-negative matrix factorization (NMF) learns a parts-based representation of faces, whereas k-means and principal components analysis (PCA) learn representations whose components are only explicable by reference to the whole. The three factorization methods were applied to a database of 2,429 greyscale facial images, each consisting of 19 × 19 pixels, forming a matrix $\boldsymbol{A} \in \mathbb{R}^{2,429 \times 19^2}$. Each method approximates $\boldsymbol{A} \approx \boldsymbol{UV}$. On the left, the $K = 49$ rows of $\boldsymbol{V}$ form a set basis images. Positive values are shown in black and negative values in red. A particular instance of a face is approximated by a weighted combination of basis images. The row of $\boldsymbol{U}$ corresponding to that image holds the coefficients for this combination, and is shown to the right of $\boldsymbol{V}$. The resulting approximation is shown on the right. Figure adapted from Lee and Seung [59].



(a) $K = 2$ (b) $K = 1$

Figure 2.14 – k-means in 2D. A datapoint is approximated by one of the K centroids that is closest to it. Assuming no duplicate datapoints, perfect reconstruction requires setting $K = N$.

estimation, with hard assignments, with respect to a mixture of Gaussians.

Consider the probability density function of a (single) multivariate Gaussian centered at $\boldsymbol{V}_k$, with identity co-variance:

$$\mathcal{P}(\boldsymbol{A}_i|\boldsymbol{V}_k) = c \exp\left(-\frac{1}{2}\|\boldsymbol{A}_i - \boldsymbol{V}_k\|^2\right) \tag{2.21}$$

where $c = \frac{1}{\sqrt{(2\pi)^M}}$. It is evident then that $-\log \mathcal{P}(\boldsymbol{A}_i|\boldsymbol{V}_k) \propto \|\boldsymbol{A}_i - \boldsymbol{V}_k\|^2$, which is the mean-squared error objective (assuming cluster indicator $\boldsymbol{U} = 1$).

The connection to a Gaussian mixture usually proceeds by defining *global* mixture weights ω_k:

$$\mathcal{P}(\boldsymbol{A}_i|\boldsymbol{V}) = \sum_k^K \omega_k \mathcal{P}(\boldsymbol{A}_i|\boldsymbol{V}_k) \tag{2.22}$$

and then introducing hard assignments as an exponent:

$$\mathcal{P}(\boldsymbol{A}_i, k|\boldsymbol{V}) = \sum_k^K \omega_k \mathcal{P}(\boldsymbol{A}_i|\boldsymbol{V}_k)^{\boldsymbol{U}_{i,k}} \tag{2.23}$$

We adopt a different view, where the mixture weights are defined *per-sample*.

$$\mathcal{P}(\boldsymbol{A}_i) = \sum_k^K \boldsymbol{U}_{i,k} \mathcal{P}(\boldsymbol{A}_i|\boldsymbol{V}_k) \tag{2.24}$$

where a hard cluster assignment collapses the sum over K to the single index where $\boldsymbol{U} = 1$. In the next chapter we will extend this view to non-negative matrix factorization, where it will be

used to derive a memorization bound.

A variant of k-means is *spherical* k*-means* [26], which only considers the angle between a centroid and a data point. Therefore instead of minimizing the mean squared-error objective, spherical k-means minimizes the *cosine dissimilarity*:

$$\text{spherical k-means}(\boldsymbol{A}, K) = \underset{\hat{\boldsymbol{A}}_K}{\arg\min} \quad \sum_i 1 - \cos(\boldsymbol{A}_i, [\hat{\boldsymbol{A}}_K]_i), \tag{2.25}$$

where $\cos(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u}^\top \boldsymbol{v}}{\|\boldsymbol{u}\|_2 \|\boldsymbol{v}\|_2}$.

Probabilistically, spherical k-means can be seen as a mixture of *von Mises–Fisher* distributions [6]:

$$\mathcal{P}(\boldsymbol{A}_i | \boldsymbol{V}_k) = c_{\kappa,M} \exp(\kappa \cdot \cos(\boldsymbol{A}_i, \boldsymbol{V}_k)) \tag{2.26}$$

where $c_{\kappa,M}$ is a constant determined by the ambient dimension M and the *concentration parameter* κ.

2.3.3 Non-negative matrix factorization (NMF)

Though many variants and constraints have been proposed, in its simplest form, non-negative matrix factorization (NMF) [60] aims to find the optimal low-rank approximation:

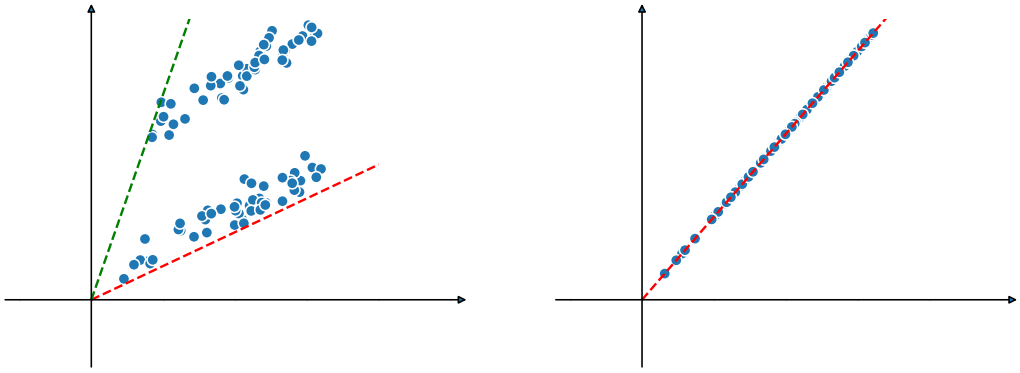
$$\begin{aligned} \text{NMF}(\boldsymbol{A}, K) = \underset{\hat{\boldsymbol{A}}_K}{\arg\min} \quad & \|\boldsymbol{A} - \hat{\boldsymbol{A}}_K\|_F^2, \\ \text{subject to} \quad & \hat{\boldsymbol{A}}_K = \boldsymbol{U}\boldsymbol{V}, \\ & \forall ij, \boldsymbol{U}_{ij}, \boldsymbol{V}_{ij} \geq 0, \end{aligned} \tag{2.27}$$

where $\boldsymbol{U} \in \mathbb{R}_+^{N \times K}$ and $V \in \mathbb{R}_+^{K \times M}$ are element-wise non-negative. The smallest K that admits an exact NMF, i.e., $\boldsymbol{A} = \hat{\boldsymbol{A}}_K$, is called the *non-negative rank* of $\boldsymbol{A}$, $rank_+(\boldsymbol{A})$.

Compared to low-rank approximation using PCA, which spans an $\mathbb{R}^k$ subspace, NMF predictions are restricted to a *simplicial cone* in the positive orthant, whose rays are the rows of $\boldsymbol{V}$. This gives NMF a probabilistic interpretation as forming a probability simplex, interpolating between the rays. In the next chapter, we adopt this view to cast NMF as a mixture model, which forms the basis of our memorization bound.

Note that although NMF minimizes a mean squared-error objective, by virtue of having $\boldsymbol{U}$ entries of any scale, it is more similar to spherical k-means than standard k-means.

Unlike PCA and k-means, NMF predictions are always non-negative. The matrix $\boldsymbol{A}$ to be



(a) $K = 2$ (b) $K = 1$

Figure 2.15 – None-negative matrix factorization in 2D. (a) Unlike PCA, NMF basis vectors are not orthogonal to each other. Furthermore, since interactions between them are only additive, they do not span $\mathbb{R}^2$, but rather a cone whose edges are the basis vectors themselves.

approximated must therefore also be non-negative. For our study of NNs, since ReLUs project their input data points onto the positive orthant, NMF can naturally be applied to network feature activations.

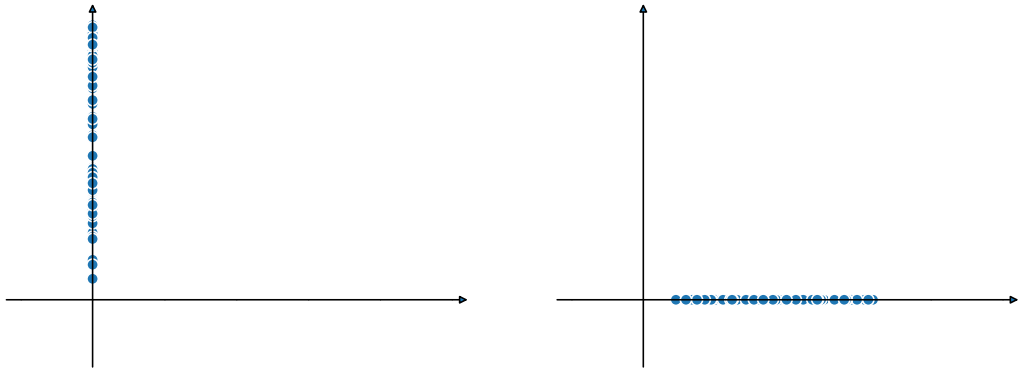
The approximated features $\hat{\boldsymbol{A}}_K$ are expressed only through additive interactions, which has been observed to lead to part-based decomposition, as shown in Figure 2.13.

NMF has gained popularity due to its producing meaningful factorizations that lend themselves to qualitative interpretation across various domains such as document clustering [101], audio source separation [40], and face recognition [41]. An example of NMF in 2D is shown in Figure 2.15. There has been work extending NMF to multiple layers [20], implementing NMF using neural networks [30] and using NMF approximations as input to a neural network [95].

In this thesis we show that, by virtue of the non-negativity of ReLU activations, NMF can be applied to CNN activations. To the best of our knowledge, the application of NMF to the activations of a neural network has not been previously proposed. The resulting factorization has applications towards both detecting memorization and network interpretability, which we demonstrate in subsequent chapters.

2.3.4 Random ablations

In Chapter 3 we compare our approach to that of Morcos et al. [66]. In their main experiment, the authors establish a positive correlation between a NN's generalization performance and its robustness to random ablations (RA) of canonical directions. RA is implemented by setting the activation value of several units or feature maps to zero, as in dropout [89].



(a) (b)

Figure 2.16 – Random ablations in 2D. Random ablation simply set a subset of $M-K$ dimensions to constant zero. This can be viewed as a crude form of dimensionality reduction. Shown are the two possible projections when setting $K = 1$.

We note that simply removing $M-K$ units from an M-dimensional datapoint results in its projection onto a K-dimensional subspace. This can be seen as a crude form of compression, i.e., by simply removing dimensions:

$$\hat{\boldsymbol{A}}_K = \boldsymbol{AV} \tag{2.28}$$

where $\boldsymbol{V} = \text{diag}(\boldsymbol{v})$ is a diagonal matrix with $\boldsymbol{v} \in \{0,1\}^M$ being a mask binary mask containing K ones and $M-K$ zeros. In accordance with the convention for dimensionality reduction, we will talk about *retaining* K basis vectors, which is the same as ablating $M-K$ dimensions. An example of RA in 2D is shown in Figure 2.16.

Note however that for PCA, k-means and NMF, retaining K dimensions with is *not* equivalent to ablating the $(M-K)$-rank approximations. In other words, RA does not consider matrix statistics, and therefore handles each datapoint independently of the others.

2.4 Conclusion

In this chapter we reviewed the fundamentals of deep learning with convolutional neural networks, and reviewed some of the practical challenges and their solutions.

We introduced the two main directions of investigation taken in this thesis, detecting memorization and interpreting network activations.

Both tasks will be addressed by use of non-negative matrix factorization. In our review of matrix factorization, we additionally reviewed several well-known methods, which we will use

as additional quantitative and qualitative baselines for our NMF-based approaches.

3 Memorization and the non-negative rank

3.1 Introduction

A fundamental challenge in machine learning for classification is that while the true objective is defined by a data distribution $(\boldsymbol{x}, y) \sim \mathcal{P}$, the objective optimized in practice is defined by an empirical distribution derived from a training set $\mathcal{D} = \{\boldsymbol{x}_i, y_i | 1 \leq i \leq N\} \sim \mathcal{P}$.

While deep NNs have achieved state-of-the-art generalization performance on many benchmarks, across various domains, they have also been shown to be over-parameterized to the point of being able to memorize very large training sets of randomly labeled data [102].

Specifically, a dataset with random labels $\tilde{\mathcal{D}} = \{\boldsymbol{x}, \tilde{y} | \boldsymbol{x} \in \mathcal{D}\}$ was created, where $\tilde{y}$ is a randomly (re-)sampled label. Interestingly, setting the probability of replacing a label with a random one to p, i.e., $\mathcal{P}(\tilde{y} = y) = 1 - p$, leads to intermediate level of overfitting and generalization. In other words, while the training error remains unaffected by label randomization (due to memorization), intermediate values of p lead to intermediate levels of test error, as can be seen in Figure 3.1.

Understanding what distinguishes networks that *learn* from networks that *memorize* is a matter of much importance, with applications towards better network design and optimization.

The empirical studies of Morcos et al. [66] and Arpit et al. [3] also study networks induced into memorization in this way. Specifically, Morcos et al. [66] set random axis-aligned directions in feature space to zero, This can be viewed as a crude form of dimensionality reduction, i.e., by simply removing canonical dimensions, as in *dropout* [89]. The authors find the network that learn and do not memorize are more robustness to this type of perturbation. In our

Some of the work presented in this chapter first appeared in [23].

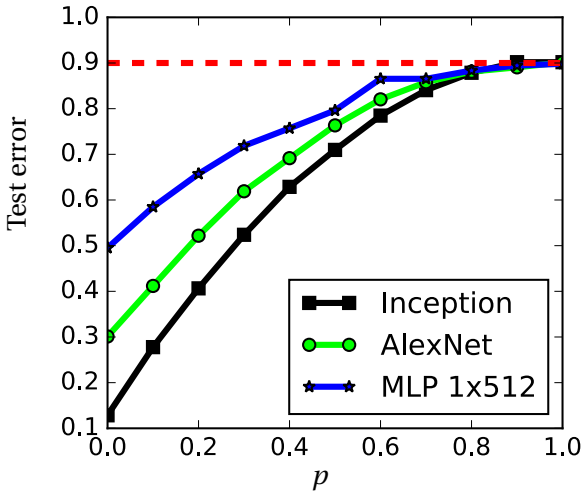


Figure 3.1 – Label randomization can be used to control the level of memorization and memorization in a CNN. Gradually increasing the probability of label randomization p results in a gradual increase in test error. Shown here are results for CIFAR-10, where in all cases the training error (not shown) is completely minimized. Figure taken from Zhang et al. [102].

experiments we refer to this method as random ablations (RA).

In this thesis we identify the key difference between networks that learn and those that memorize as *robustness to compression*. We begin our study of factors involved in memorization by recalling the information bottleneck principle (IB) [86]. IB proposes that *compression* in hidden layers plays a key role in good generalization.

Let X and Y be random variables that represent the network input and output, respectively, and let Z represent a hidden layer. The IB-functional defines an optimality criterion for hidden activations:

$$\mathcal{I}(Y,Z) - \beta\mathcal{I}(X,Z) \tag{3.1}$$

where $\mathcal{I}(\cdot,\cdot)$ is the *mutual information* and β controls the level of compression.

To make the problem of memorization explicit, we modify the IB-functional compression term:

$$\mathcal{I}(Y,Z) - \beta\mathcal{I}(\mathbf{i},Z) \tag{3.2}$$

where the random variable $\mathbf{i}$ indexes the training set $\mathcal{D}$, taking on values in $[1,\cdots,N]$ with equal probability. For networks with the same value of $\mathcal{I}(Y,Z)$, lower $\mathcal{I}(\mathbf{i},Z)$ directly indicates less memorization and implies better generalization. A similar expression was used in Alemi

et al. [1], albeit for unsupervised learning.

We study the term $\mathcal{I}(\mathbf{i}, Z)$ by considering a related concept, the *common information* [98; 100], which upper bounds the mutual information between a pair of variables via an auxiliary variable that explicitly captures interactions between the pair. This allows us to both quantify memorization, as well as analyze a network's hidden representations through analysis of the auxiliary variable, which is the basis for the subsequent chapter in this thesis.

Inspired by [14], we prove that an exact non-negative factorization of a ReLU activation matrix contains the common information between $\mathbf{i}$ and Z, yielding an upper bound on memorization in terms of the *non-negative rank* (Section 3.2). We further show a the non-negative rank to be a natural measure of complexity for a ReLU activation matrix capturing its *departure from linearity* (Section 3.3).

Although computing the non-negative rank is NP-hard [93], we can restrict it with approximate non-negative matrix factorization (NMF). Consequently, we propose to estimate the impact of NMF on NN performance over a grid of approximation ranks K, as described in Section 3.4.

This procedure can be seen as measuring the robustness of a NN to increasing compression applied to its activations. Throughout our experiments in Section 3.5, we therefore compare our NMF approach to three additional dimensionality reduction techniques, namely principal component analysis (PCA), k-means, and RA.

Our experimental setup is similar to that of Morcos et al. [66] in that both NMF and RA are a form of compression to hidden activations. Our results show that robustness to NMF compression is much more correlated with low memorization/high generalization than robustness to RA.

Using our approach, we make several interesting observations, which we verify over a variety of network architectures trained on several image and audio datasets. In Section 3.5.2, we show that the structure of convolutional networks results in *memorization being localized to deeper layers*.

Furthermore, we show our method can effectively detect memorization during training (Section 3.5.3), as well as with respect to individual output classes (Section 3.5.4).

3.2 Memorization bound through Common information

The *common information* [98; 100] between random variables A and B is:

$$\mathcal{C}(A,B) \triangleq \inf_{F:A\perp B|F} \mathcal{I}((A,B),F) \tag{3.3}$$

where (A, B) represents the joint distribution of A and B. A variable F that satisfies $\mathcal{I}(AB|F) = 0$ is said to contain the common information between A and B, and by definition $\mathcal{I}((A, B), F) \geq \mathcal{C}(A, B)$.

Since $\mathcal{I}(A, B) \leq \mathcal{C}(A, B)$, to obtain an upper bound over the mutual information it is sufficient to find an appropriate variable F. This is the strategy taken by Braun and Pokutta [14], who prove the following proposition:

Proposition 1 (Braun and Pokutta [14])**.** *Let A and B be discrete random variables over a finite set, and let their joint distribution be represented as a non-negative matrix $\boldsymbol{A} \in \mathbb{R}_+^{|A|\times|B|}$ where $\mathcal{P}(A = i, B = j) = \boldsymbol{A}_{ij}$. Let $\boldsymbol{A} = \boldsymbol{UV}$ be an exact NMF of rank K. Then:*

$$\mathcal{I}(A, B) \leq \log K \tag{3.4}$$

Proof. NMF can be seen as a sum of K rank-1 matrices:

$$\boldsymbol{A} = \sum_k^K \boldsymbol{U}_{\cdot k}\boldsymbol{V}k\cdot \tag{3.5}$$

$$\tag{3.6}$$

Define a discrete random variable F taking on values in $\{1, \cdots, K\}$, such that:

$$\mathcal{P}(F = k|A = i, B = j) = \frac{\boldsymbol{U}_{i,k}\boldsymbol{V}_{k,j}}{\boldsymbol{A}[i, j]} \tag{3.7}$$

It then follows that:

$$\mathcal{P}(F = k) = \sum_{i,j} \boldsymbol{U}_{i,k}\boldsymbol{V}_{k,j} \tag{3.8}$$

$$\mathcal{P}(A = i, B = j|F = k) = \frac{\boldsymbol{U}_{i,k}\boldsymbol{V}_{k,j}}{\sum_{i',j'} \boldsymbol{U}_{i',k}\boldsymbol{V}_{k,j'}} \tag{3.9}$$

and we can already see that the probability consists of a product of terms that depend either on i or on j, which verifies the conditional independence of A and B given F. Finally, by using the definition of common information and the properties of mutual information:

$$\mathcal{I}(A, B) \leq \mathcal{C}(A, B) \leq \mathcal{I}((A, B), F) \leq \mathcal{H}(F) \leq \log K \tag{3.10}$$

where H is Shannon entropy. □

It is interesting to seek a similar interpretation of NN activation matrices. While the index variable **i** is discrete and can be assumed to distribute uniformly across the rows, the variable

Z is not discrete. Instead, Z is a *continuous vector* variable, and as a result, unlike Proposition (1), the individual entries of a ReLU activation matrix $\boldsymbol{A}$ (even after normalization) cannot be interpreted as probabilities.

Fortunately, we can derive similar bounds also in this case, based on a probabilistic interpretation of clustering.

Proposition 2. *Let the $\boldsymbol{A} \in \mathbb{R}_+^{N\times M}$ represent the joint distribution of a discrete index variable $\mathbf{i}$, uniformly distributed across the N rows, and a continuous vector variable Z, such that:*

$$\begin{aligned}\mathcal{P}(\mathbf{i} = i, Z = \boldsymbol{z}) &= \mathcal{P}(\mathbf{i})\mathcal{P}(Z|\mathbf{i}) \\ &= \frac{1}{N}\varphi(\boldsymbol{z}|\boldsymbol{A}_i)\end{aligned} \tag{3.11}$$

where φ is a probability density function defined below.

Let $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V}$ be an exact matrix factorization of rank K, where $\boldsymbol{U}$ is non-negative (but $\boldsymbol{V}$ may not be). Then:

$$\mathcal{I}(\mathbf{i}, Z) \leq \log K \tag{3.12}$$

Proof. Following the discussion in Section 2.3, the matrix factorization defines *per-row i* mixture of probability distributions centered as the rows of $\boldsymbol{V}$, with coefficients $\frac{\boldsymbol{U}_{i\cdot}}{\sum_k \boldsymbol{U}_{ik}}$ as mixing weights.

We thus define $\varphi(\boldsymbol{z}|\boldsymbol{a})$ to be the probability of $\boldsymbol{z}$ under the mixture defined by $\boldsymbol{a}$.

For $\boldsymbol{a} \notin \boldsymbol{A}$, let $U(\boldsymbol{a})$ be the operator to extract optimal coefficients with respect to $\boldsymbol{V}$, considering $\boldsymbol{V}$ as a *frozen* basis. The operator is a simple convex program that can be solved deterministically. It is of course the case that $U(\boldsymbol{A}_i) = \boldsymbol{U}_{i\cdot}$.

We therefore have:

$$\varphi(\boldsymbol{z}|\boldsymbol{a}) = \sum_k^K \frac{U(\boldsymbol{a})_k}{\sum_{k'} U(\boldsymbol{a})_{k'}} \phi(\boldsymbol{z}|\boldsymbol{V}_{k\cdot}) \tag{3.13}$$

where ϕ is any density function over $\mathbb{R}^M$, e.g., a von Mises–Fisher distribution with fixed κ.

The first term in the above sum indicates $\mathcal{P}(F = k|\mathbf{i} = i)$. The second term in the sum indicates $\mathcal{P}(Z = \boldsymbol{z}|F = k)$.

It then follows that:

$$\mathcal{P}(\mathbf{i}=i, Z=z) = \frac{1}{N}\varphi(\boldsymbol{z}|\boldsymbol{A}_i) \tag{3.14}$$

$$= \mathcal{P}(\mathbf{i}=i)\sum_{k}^{K}\mathcal{P}(F=k|\mathbf{i}=i)\mathcal{P}(Z=z|F=k) \tag{3.15}$$

and we get that Z is conditionally independent of $\mathbf{i}$ given F. □

This proof builds on a **clustering interpretation of matrix factorization** which is obvious for k-means, but perhaps less intuitive for NMF. It does not apply to PCA, since the matrix $\boldsymbol{U}$ typically contains negative values, and the mixing weights in Eq. (3.13) depend on the non-negativity of factorization coefficients.

The mixing weights reflect properties of the factorization. For k-means, for instance, cluster assignments are hard, i.e. the probability mass is concentrated on a single value k^* where $U(\boldsymbol{a})_{ik^*} = 1$. The density in this case collapses to $\varphi(\boldsymbol{z}|\boldsymbol{A}_i) = \phi(\boldsymbol{z}|\boldsymbol{V}_{k^*\cdot})$.

With NMF, the factorization defines a *simplex*, on which each point defines a categorical probability distribution over the K corners of the simplex, i.e., the rows of $\boldsymbol{V}$. Conditioning on a vector in this case, $\varphi(\cdot|\boldsymbol{a})$, results in a soft mixture. This view explicitly casts NMF as a relaxation of spherical k-means clustering. In other words, while NMF does indeed 'span' a cone in $\mathbb{R}^M$, perfectly reconstructing all points in the cone, the probability density defined above considers the rays of the NMF cone as modes and *penalizes mode averaging*. That is, for a point $\boldsymbol{a}$ inside the simplex, the mode of $\varphi(\cdot|\boldsymbol{a})$ is not at $\boldsymbol{a}$, but rather at the ray closest to it.

The smallest K that satisfies the bound is the non-negative rank of $\boldsymbol{A}$, yielding:

Corollary 2.1. *With* $\mathbf{i}$, Z *and* $\boldsymbol{A}$ *defined as in Proposition (2):*

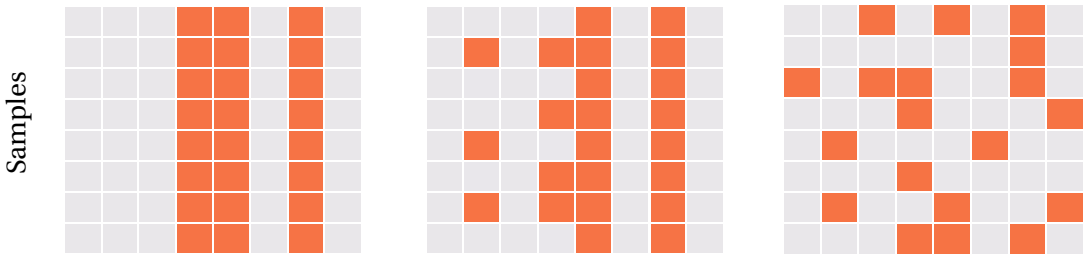
$$\mathcal{I}(\mathbf{i}, Z) \leq \log rank_+(\boldsymbol{A}) \tag{3.16}$$

In the next section we show an additional view of $\text{rank}_+(\boldsymbol{A})$ as a measure of non-linearity with respect to ReLU activation matrices.

3.3 Non-linearity and rectangle cover number

Consider a ReLU layer parameterized by a weight matrix $\boldsymbol{W} \in \mathbb{R}^{M\times P}$. For a batch of N inputs $\boldsymbol{X} \in \mathbb{R}^{N\times M}$, we compute the layer activation matrix $\boldsymbol{A}$ as $\boldsymbol{A} = \max(\boldsymbol{XW}, 0) \in \mathbb{R}_+^{N\times P}$. We omit the bias term for notational convenience.

Since ReLU is piece-wise linear, the processing of a single input $\boldsymbol{x}$ by a ReLU network is



(a) Linear $rc(\boldsymbol{M}) = 1$ (b) Approximately linear $rc(\boldsymbol{M}) = 3$ (c) Highly non-linear $rc(\boldsymbol{M}) = 8$

Figure 3.2 – The support of a ReLU activation matrix is determined by the threshold at zero. (a) When all the rows of the support are identical, there is a sub-weight-matrix such that the layer is fully linear with respect to the input batch. (b, c) As the support becomes more complex, which we characterize by the increase in its rectangle cover number, the layer becomes more non-linear.

equivalent to sampling a *linear* sub-network with respect to the sample [97]. The linear function is obtained by setting to zero the columns of each $\boldsymbol{W}$ whose dot product with the input is negative (and would thus be set to zero by ReLU), after which the ReLU can be removed.[1]

Extending this notion to a batch of several input samples, suppose that at some layer the samples are sufficiently close such that they all share the same ReLU mask $\boldsymbol{m} \in \{0,1\}^P$. In this case, we may say that *the layer is linear with respect to its input batch*. This is because, for the entire batch, instead of using ReLU, we could zero out a subset of columns and obtain a linear system, i.e., $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{W}\text{diag}(\boldsymbol{m})$.

To characterize how far an activation matrix diverges from the linear case, consider the *support* $\boldsymbol{M} = \text{supp}(\boldsymbol{A})$, such that $\boldsymbol{M}_{i,j} = 1$ where $\boldsymbol{A}_{i,j} > 0$ and is 0 elsewhere. Because $\boldsymbol{A}$ is a ReLU activation matrix, the support is mainly determined by the thresholding at zero.[2] If all the rows of $\boldsymbol{M}$ are identical to a unique vector $\boldsymbol{m}$, we can say the layer is completely linear with respect to $\boldsymbol{X}$. In general, the 'simpler' the support $\boldsymbol{M}$, the closer to linearity the layer.

One measure that captures this idea is the *rectangle cover number* of a matrix, $rc(\boldsymbol{M})$, an important quantity in the study of communication complexity [53]. Also known as the *Boolean rank*, $rc(\boldsymbol{M})$ is the smallest number r for which there exist binary matrices $\boldsymbol{U}_B \in \{0,1\}^{n\times r}$, $\boldsymbol{V}_B \in \{0,1\}^{r\times q}$ such that their *Boolean* matrix multiplication satisfies $\boldsymbol{M} = \boldsymbol{U}_B\boldsymbol{V}_B$. As a complexity measure for ReLU activations, $rc(\boldsymbol{M}) = 1$ means the layer is linear with respect to its input, and

[1] This is a similar intuition to that of viewing dropout as an approximation to a model ensemble, where the dropout mask is seen to sample a sub-network [89].

[2] The probability of an activation value being exactly zero prior to thresholding is negligible.

higher values $rc(\boldsymbol{M})$ imply increasing non-linearity. This is visualized in Figure 3.2.

Intuitively, imagine having to fit a layer with 'ReLU switches', each of which controls a subset of weight matrix columns. In the linear case, one switch would suffice to describe the data. In the most non-linear case, we would require a switch for every column, which is also the maximal value of $rc(\boldsymbol{M})$. Since adding a new row to the support $\boldsymbol{M}$ such that the row is a union of existing rows of $\boldsymbol{M}$ does not increase the rectangle cover number, there is no increase in the number of switches.

The non-negative rank is also hard-constrained by the combinatorial arrangement of $\text{supp}(\boldsymbol{A})$, but additionally accounts for the precise value in the non-zero entries of $\boldsymbol{A}$, thus yielding [33]:

$$rc(\text{supp}(\boldsymbol{A})) \leq \text{rank}_+(\boldsymbol{A}) \tag{3.17}$$

3.4 Estimating the non-negative rank

For convolutional networks, we reshape the activation *tensor* from $N \times C \times H \times W$ to $(N \cdot H \cdot W) \times C$, i.e., we flatten the batch (N) and spatial (H, W) dimensions to form an activation matrix with C columns, where C is the number of channels in that layer. We then inversely reshape the factorized features to continue forward propagation. Compression is thus applied treating each single C-dimensional vector as a separate data point, rather than each whole feature map. In the convolutional case, the index variable $\mathbf{i}$ therefore indexes *patches*, rather than whole images.

Although computing $\text{rank}_+(\boldsymbol{A})$ is NP-hard, we *restrict it* by performing approximate NMF. The factorization error depends on the magnitude of the activations, which makes comparison across networks and layers difficult. We therefore evaluate the impact of compression by forward-propagating the compressed activations and measuring the change in final network performance (i.e., classification accuracy) as we change K.

Concretely, if we let $\boldsymbol{A}_j$ be the activation matrix at layer j, during the forward pass we replace the feature activations of one or several layers with their rank K approximations:

$$\begin{aligned} \tilde{\boldsymbol{A}}_j &= \boldsymbol{U}\boldsymbol{V} \\ \boldsymbol{A}_{j+1} &= \Lambda_{j+1}\left(\tilde{\boldsymbol{A}}_j\right) \end{aligned} \tag{3.18}$$

The resulting curves allow us to estimate a value for $\text{rank}_+(\boldsymbol{A})$. We estimate the usefulness of our approach by applying it to networks with varying levels of memorization and generalization error.

3.4.1 Single-class batches

The activation matrix $\boldsymbol{A}$ represents a *batch* of samples from a larger training set. The distribution of Y over the batch defines a lower bound on the memorization term $\mathcal{I}(\mathbf{i}, Z)$:

Proposition 3. *With* $\mathbf{i}$*,* Z*, and* Y *defined as above,* $\mathcal{I}(\mathbf{i}, Z) \geq \mathcal{I}(Z, Y)$

Proof. The chain rule of mutual information decomposes:

$$\begin{aligned} \mathcal{I}(Z,(i,Y)) =& \mathcal{I}(\mathbf{i},Z) + \mathcal{I}(Z,Y|\mathbf{i}) \\ =& \mathcal{I}(Z,Y) + \mathcal{I}(\mathbf{i},Z|Y) \end{aligned} \tag{3.19}$$

Dependence of $\mathbf{i}$ is described by the Markov chain

$$\begin{aligned} &\mathbf{i} \rightarrow Y \\ &\mathbf{i} \rightarrow X \rightarrow Z \end{aligned} \tag{3.20}$$

and we get that $\mathcal{I}(Z, Y|\mathbf{i}) = 0$. Since $\mathcal{I}(\mathbf{i}, Z|Y) \geq 0$ we get the result. □

We thus sample a batch where all input samples map to same output, such that $\mathcal{I}(Z, Y) = 0$ and $\mathcal{I}(\mathbf{i}, Z)$ is lower bounded only by zero. We refer to such batches as *single-class* batches, as opposed to *multi-class* batches which are sampled i.i.d. The effect of different sampling is discussed in Section 3.5.2.

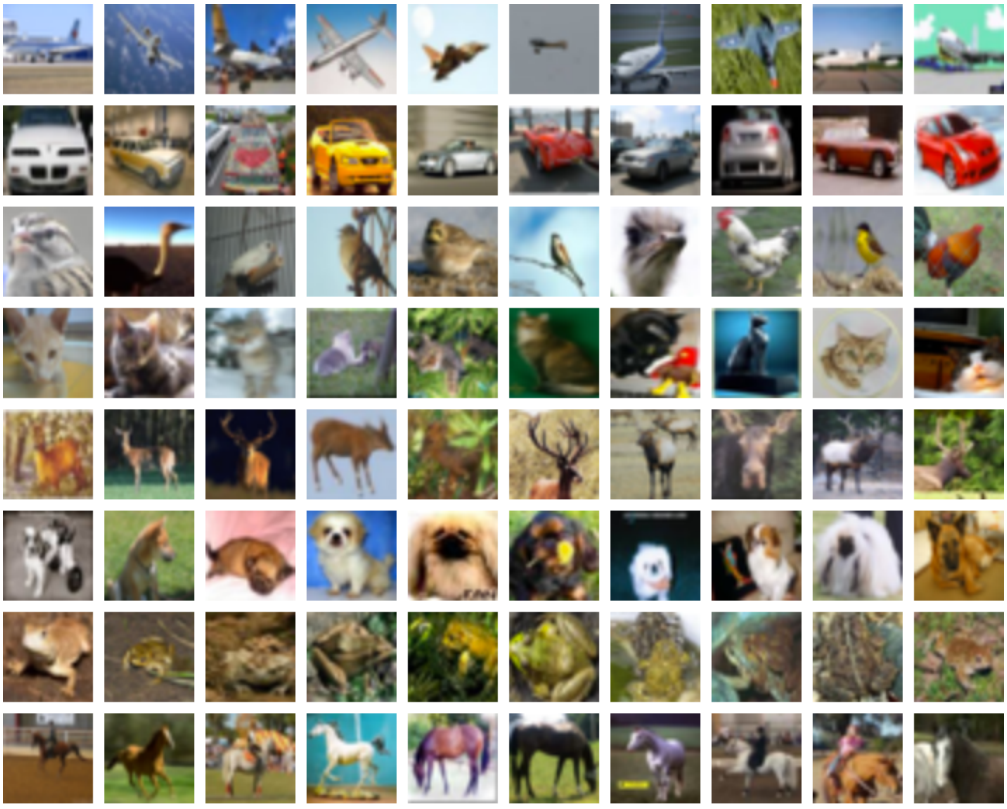
3.5 Experiments

In this section we present results that empirically confirm the connection of the non-negative rank to memorization and generalization. Our experiments include various datasets and network architectures, and check for correlation between the non-negative rank of deep activation matrices and memorization/generalization post-training, during training, as well as per-class.

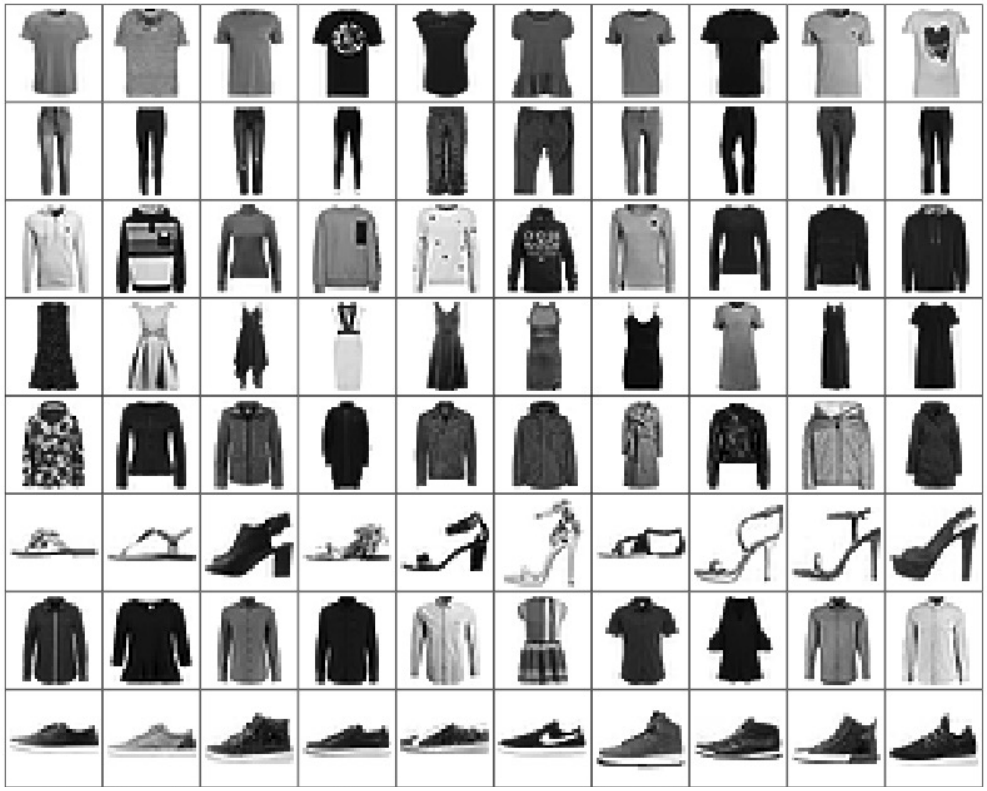
3.5.1 Datasets and networks

Datasets

We perform experimental evaluations on several image datasets (see examples in Figure 3.2), as well as an audio classification dataset.

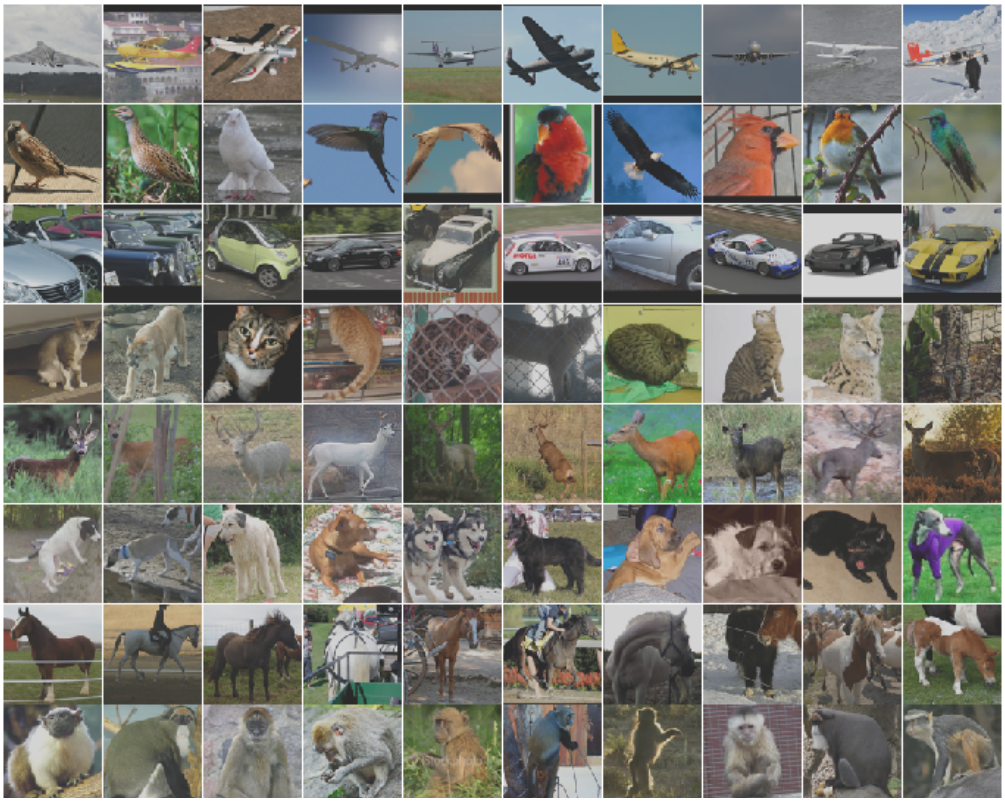


(a) CIFAR-10 examples



(b) Fashion-MNIST examples

(c) SVHN examples



(d) ImageNet examples

Figure 3.2 – Examples from the four image datasets we used in our study of NN memorization.

CIFAR-10 CIFAR-10 [54] consists of 60K 32×32 RGB images classified into ten categories: *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship*, *truck*. The dataset has a standard split into 50K training images and 10K test images.

Fashion-MNIST Fashion-MNIST [99] consists of 70K 28×28 greyscale images, classified into ten categories: *T-shirt/top*, *trouser*, *pullover*, *dress*, *coat*, *sandal*, *shirt*, *sneaker*, *bag*, *ankle boot*. The dataset has a standard split into 60K training images and 10K test images.

SVHN SVHN [67] (Street View House Numbers) consists of approximately 560K 32×32 RGB images classified into ten categories, one for each digit 0-9. The images are cropped from Google Street View images containing house numbers.

The dataset has a standard split into approximately 73K training images, 25K test images, and an additional 531K 'easy' samples as extra training images. In this thesis, we only make use of the first training set.

ImageNet ImageNet [80] is a dataset of about 1.2M RGB images classified into 1K categories, including types of animals, flowers, furniture, tools, etc. While these images are originally of varying sizes, we use them here in conjunction with a pre-trained VGG network [87], for which the input was scaled to 224×224.

Urban Sounds Urban Sounds [82] consists of almost 9K audio clips, each of up to 4 seconds long, classified into ten categories: *air conditioner*, *car horn*, *children playing*, *dog bark*, *drilling*, *engine idling*, *gun shot*, *jackhammer*, *siren*, *street music*. The dataset has a standard split into 10 folds, of which we use the first 8 for training and the final 2 as test.

Neural Network Architectures

The exact architectures we used for each dataset are given in Table 3.1. We denote a linear or convolutional layer followed by a ReLU as Linear_+ and Conv_+, respectively.

In addition to these networks, we tested VGG-19 on ImageNet data.

3.5.2 Feature compression and memorization

We study networks that have been forced into different levels of memorization due to label randomization applied to their training set [102], as described in Section 3.1. The level of induced memorization is controlled by setting a probability p for a training label to be ran-

CIFAR-10

Type	Dim	Kernel	Padding	Stride
Conv_+	64	3	1	1
Conv_+	64	3	1	1
Conv_+	128	3	1	2
Conv_+	128	3	1	1
Conv_+	128	3	1	1
Conv_+	256	3	1	2
Conv_+	256	3	1	1
Conv_+	256	3	1	1
Conv_+	512	3	1	2
Conv_+	512	3	1	1
Conv_+	512	3	1	1
Linear	10	-	-	-

Urban Sounds

Type	Dim	Kernel	Padding	Stride
Conv_+	64	3	1	1
MaxPool	-	2	-	1
Conv_+	128	3	1	1
Conv_+	128	3	1	1
MaxPool	-	2	-	1
Conv_+	256	3	1	1
Conv_+	256	3	1	1
MaxPool	-	2	-	1
Conv_+	512	3	1	1
Conv_+	512	3	1	1
MaxPool	-	2	-	1
Linear_+	4096	-	-	-
Linear_+	4096	-	-	-
Linear	10	-	-	-

SVHN

Type	Dim	Kernel	Padding	Stride
Conv_+	64	3	1	1
Conv_+	64	3	1	1
Conv_+	128	3	1	2
Conv_+	128	3	1	1
Conv_+	256	3	1	2
Conv_+	256	3	1	1
Conv_+	512	3	1	2
Conv_+	512	3	1	1
Linear	10	-	-	-

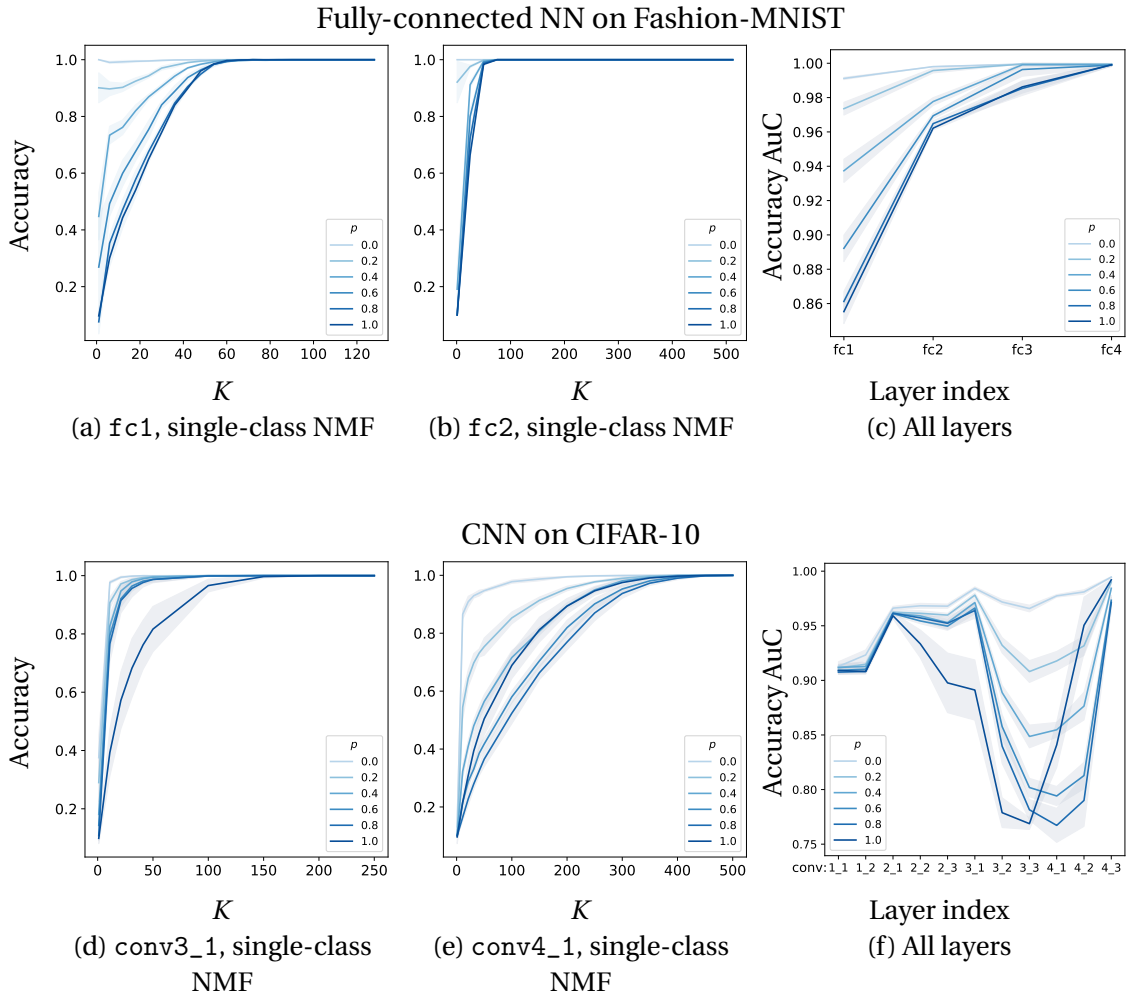
Fashion-MNIST

Type	Dim	Kernel	Padding	Stride
Linear_+	128	-	-	-
Linear_+	512	-	-	-
Linear_+	2048	-	-	-
Linear_+	2048	-	-	-
Linear	10	-	-	-

Table 3.1 – Neural network architecture used for each dataset in this chapter.

domized, i.e., $p = 0$ is the unmodified dataset and $p = 1$ has fully random labels. The capacity of these networks is sufficiently large that they achieve a **training accuracy of 1 in all cases**, for all values of p.

As such, we use batches of training data and observe the accuracy drop, from 1 to constant guess, as the level of compression is increased. In all experiments, sampling single-class batches is done with respect to the label used for training (i.e., the random label if $p > 0$). Since all datasets we consider here have 10 classes, we stochastically sample 10 batches, one per class in the single-class case, and standard i.i.d in the multi-class case. We have found all methods discussed below to be robust to the batch size (e.g., 20-100) and the exact samples chosen. In all our experiments we set the batch size to 50.



(a) `fc1`, single-class NMF (b) `fc2`, single-class NMF (c) All layers

(d) `conv3_1`, single-class NMF (e) `conv4_1`, single-class NMF (f) All layers

Figure 3.3 – Layer-by-layer NMF compression in (a,b,c) fully-connected vs. (d,e,f) CNNs. Increasing p indicates a higher level of memorization. We compress the activations of a layer using NMF with increasing K and observe the impact on classification performance. In (a,b,d,e) we see that layers at different depth respond differently. (c,f)To get a global view of the effect of compression, we describe every layer by the area under the curve (AuC) of its K vs. classification accuracy curve. Unlike fully-connected networks, memorization in CNNs is localized to deeper layers. Interestingly, the case $p = 1$ shifts the process to earlier layers.

Layer by Layer Analysis

We begin by studying each NN layer individually using NMF. We trained a total of 60 neural networks, ten networks (with different random initializations) trained per randomization level p.

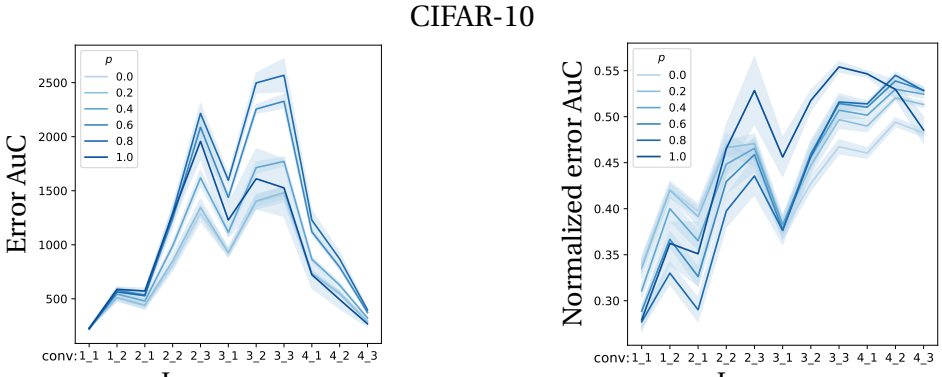
In Figure 3.3 (a,b,d,e) we show examples of NMF compression applied to individual layers

of these networks, where we measured the impact on classification accuracy as we vary the factorization rank K. As can be seen, networks trained with different levels of label randomization respond differently to compression, depending on the layer and overall architecture. At some layers, memories with no induced memorization are significantly more robust to NMF compression than those forced into memorization.

The profile of such a K-vs.-accuracy curve can be characterized by its area under the curve (AuC), such that a *higher* non-negative rank memorization corresponds with *lower* AuC, and vice versa. To compare between layer, we normalize the AuC by dividing it by the layer width, such that the AuC is at most 1. Considering the AuC allows us to characterize each network layer with a single scalar, giving a bird's eye view of the network, shown in Figure 3.3 (c,d).

In the fully-connected networks shown in Figure 3.3c, we find that deeper layers are more robust to compression than earlier layers, as expected if we assume deeper activations become more abstract and invariant to nuisance variables. In the convolutional networks, however, shown in Figure 3.3f, we find that memorization is localized to deeper layers, where there is a big difference between memorization levels.

Early layers and the last layer show similar statistics across all memorization levels. We hypothesize that this is due to the network shifting from "place-coding" to "channel-coding" (similar observations are made in [81]), as features extracted at lower layer are integrated to produce more global representations. Interestingly, setting $p = 1$ shifts the process to earlier layers, explaining why layer-by-layer these networks appear as outliers.



(a) Raw NMF error

(b) Normalized NMF error

Figure 3.4 – Layer-by-layer view of (a) raw and (b) normalized NMF reconstruction errors. This is the error that the NMF objective is trying to minimize. The reconstruction error itself is sensitive to the arbitrary magnitude of network weight, which makes it difficult to interpret across layer and networks.

In addition to classification accuracy, the NMF reconstruction error itself is also a quantity of interest. The main difficulty involved with interpreting the NMF error is scale. The error depends on the magnitude of the activations, which varies across networks, layers, and even channels.

In Figure 3.4 (a) and (b) we show the raw and normalized NMF reconstruction error, i.e. $\|\boldsymbol{A} - \tilde{\boldsymbol{A}}\|_2$ and $\frac{\|\boldsymbol{A}-\tilde{\boldsymbol{A}}\|_2}{\|A\|_2}$ respectively. Measurements are taken over the same networks discussed in Figure 3.3 (d,e,f). Observing the normalized values reveals that, proportionally, activation matrices become harder to approximate with depth, with an interesting interaction between the memorization level and depth. The error in absolute terms echo the accuracy curve, with $p = 1$ again presenting outlier behavior.

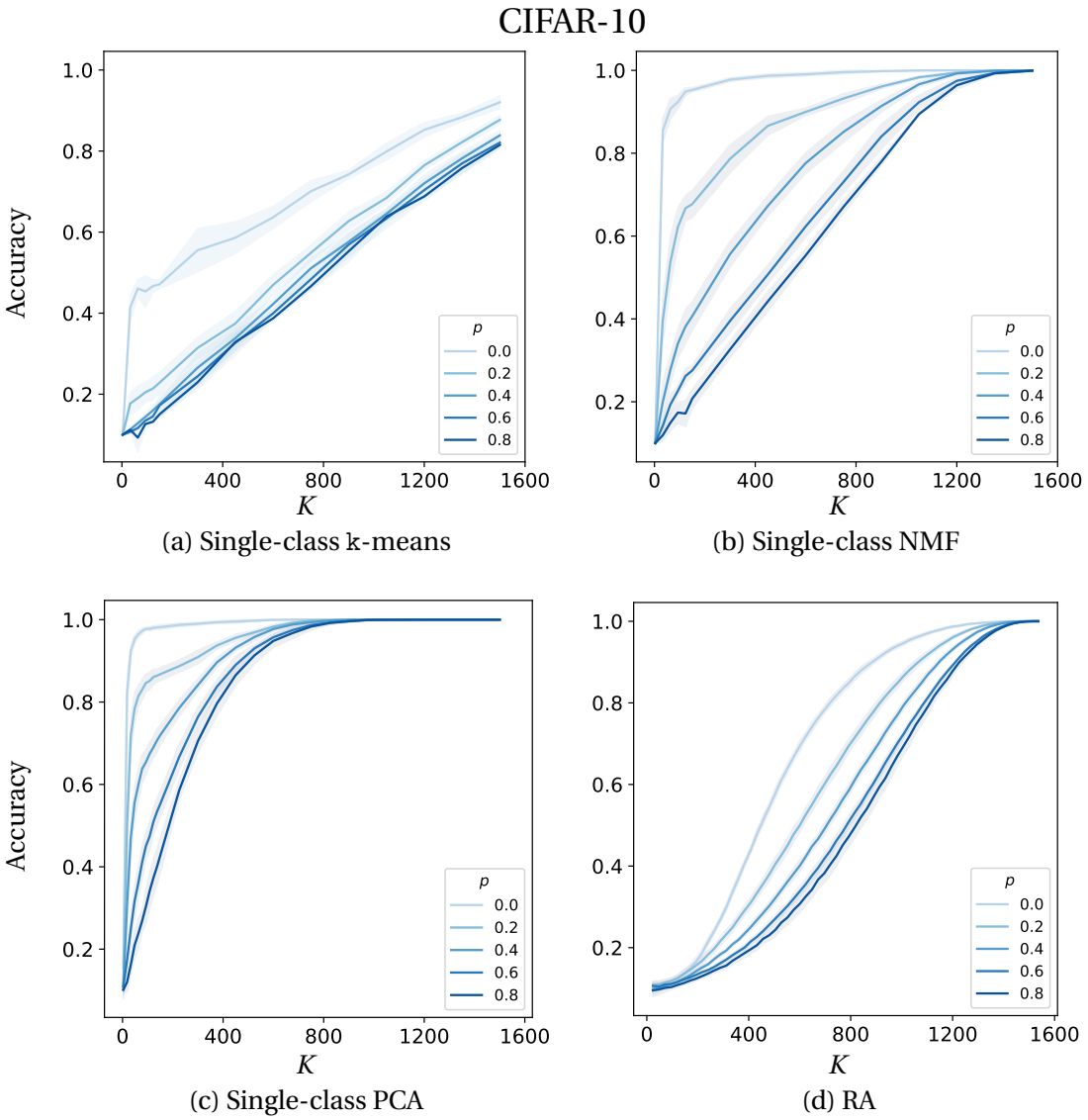
Factorization methods

We compare `k`-means, PCA, NMF and RA. Rather than compress a single layer, we sequentially apply compression to several layers one after another during the forward pass. In particular, we apply compression to each layer in the final convolutional block of our CNN, consisting of three layers, each of which consists of 512 channels (see Table 3.1). In fully-connected networks, we applied compression to all layers.

In Figure 3.5 we report results for the CIFAR-10 dataset. Given its limited expressiveness, `k`-means requires a very large K to approach good classification accuracy, and the result is not useful for distinguishing levels of memorization. NMF produces the best results, clearly distinguishing different levels of $\mathcal{I}(\mathbf{i}, Z)$. In contrast, PCA, which is less constrained, regains good accuracy already with small values of K, but is less discriminative with respect to the level of memorization.

Finally, we confirm that robustness to RA correlates with less memorization, however less so than NMF. It should be noted, however, that NMF does show more variance than other methods, and incurs some additional overhead, as discussed below.

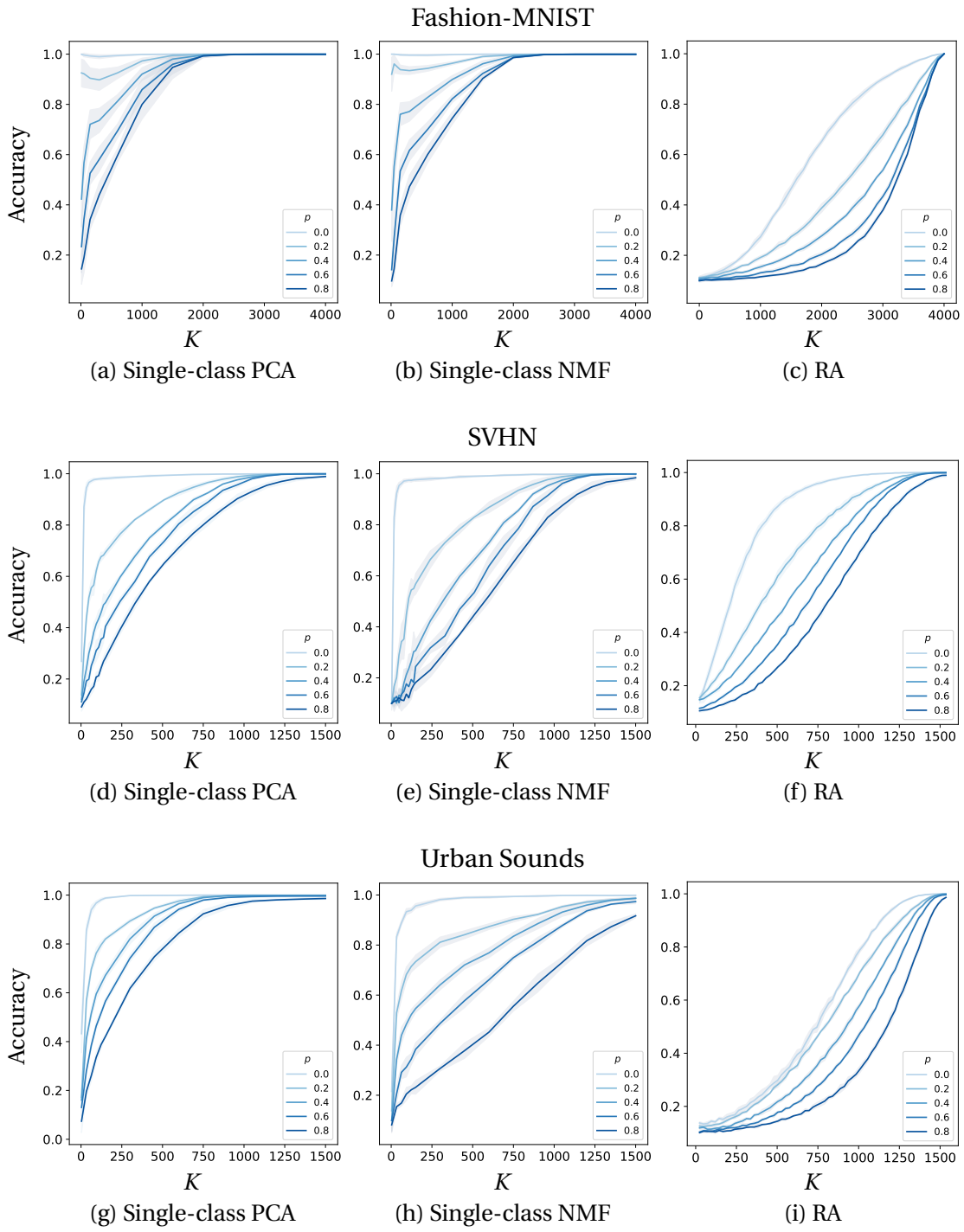
We show additional results in Figure 3.6 for on three additional datasets and network architectures, including a fully-connected network for Fashion-MNIST. These results further establish NMF as a good method for distinguishing levels of memorization.



(a) Single-class k-means (b) Single-class NMF

(c) Single-class PCA (d) RA

Figure 3.5 – We compress NN activation using various matrix factorization methods. Factorization was applied to the final three (convolutional) layers of CNNs trained on CIFAR-10 with different levels of label randomization (p). (a) k-means requires on the order of N (number of samples) centroids to approach a good approximation, and does not seem to be a feasible method for detecting memorization. (b) NMF approximates the activations well enough to retain good accuracy, while clearly distinguishing between different levels of $\mathcal{I}(\mathbf{i}, Z)$ across the networks. (c) Due to its expressiveness, PCA is able to well approximate the activations even with small values of K, but is therefore less distinctive between levels of $\mathcal{I}(\mathbf{i}, Z)$. (c) Though not taking into account batch statistics, RA does distinguish between different levels of memorization, albeit less significantly.



(a) Single-class PCA (b) Single-class NMF (c) RA

(d) Single-class PCA (e) Single-class NMF (f) RA

(g) Single-class PCA (h) Single-class NMF (i) RA

Figure 3.6 – We show that NMF-based compression is sensitive to memorization in diverse settings. Each row shows results for a specific dataset and network architecture. PCA and RA consistently show less sensitivity to memorization compared with NMF.

Following the discussion in Section 3.4.1, in Figure 3.7 we evaluated the effect of compression with multi-class batches, i.e., sampled i.i.d. Compared to single-class batches (Figures 3.5 and 3.6), the different levels of memorization are indeed less distinct. Another view of this behavior is that batching samples from various classes together assures the batch distribution is multi-modal. This naturally requires a higher factorization rank in order to obtain a good approximation.

Ablating NMF and PCA directions

We study the impact of *ablating* the activation in the directions found by NMF and PCA by forward propagating the residual, i.e.,

$$\boldsymbol{A}_{j+1} = \Lambda_{j+1}\left(\boldsymbol{A}_j - \tilde{\boldsymbol{A}}_j\right) \tag{3.21}$$

This is interesting because in the case of PCA, for instance, the top K directions are those that capture most of the variance in the activation matrix, and presumably the K directions found by NMF are of similar importance. This is not true for RA, where the ablated directions are of no special importance.

In Figure 3.8 we see that networks with no induced memorization that are *most vulnerable* to ablation of NMF and PCA direction. In other words, while non-memorizing networks are more robust to *random* ablations, they are less robust to ablations of specific important directions. This is in contrast to the interpretation of Morcos et al. [66] that non-memorizing networks are more robust to ablations of single directions.

3.5.3 Feature compression and generalization

So far we have dealt with networks that were induced into memorization by randomizing their training labels. We now show that NMF is useful for predicting good *generalization* in a more realistic setting.

For this experiment, we trained 96 CNNs on CIFAR-10, over a grid of hyper-parameter values for the batch size, weight decay and optimization algorithm, SGD vs. ADAM [52]. Following the procedure of Section 3.5.2, we computed the AuC of the K-vs.-accuracy curve of each network's final convolutional block.

In Figure 3.9 we compare the AuC of NMF, PCA, and RA against the generalization error on the **test** set. While all three methods show correlation with generalization error, NMF is most correlated with a Pearson correlation of -0.82, followed by PCA with -0.64 and RA with -0.61.

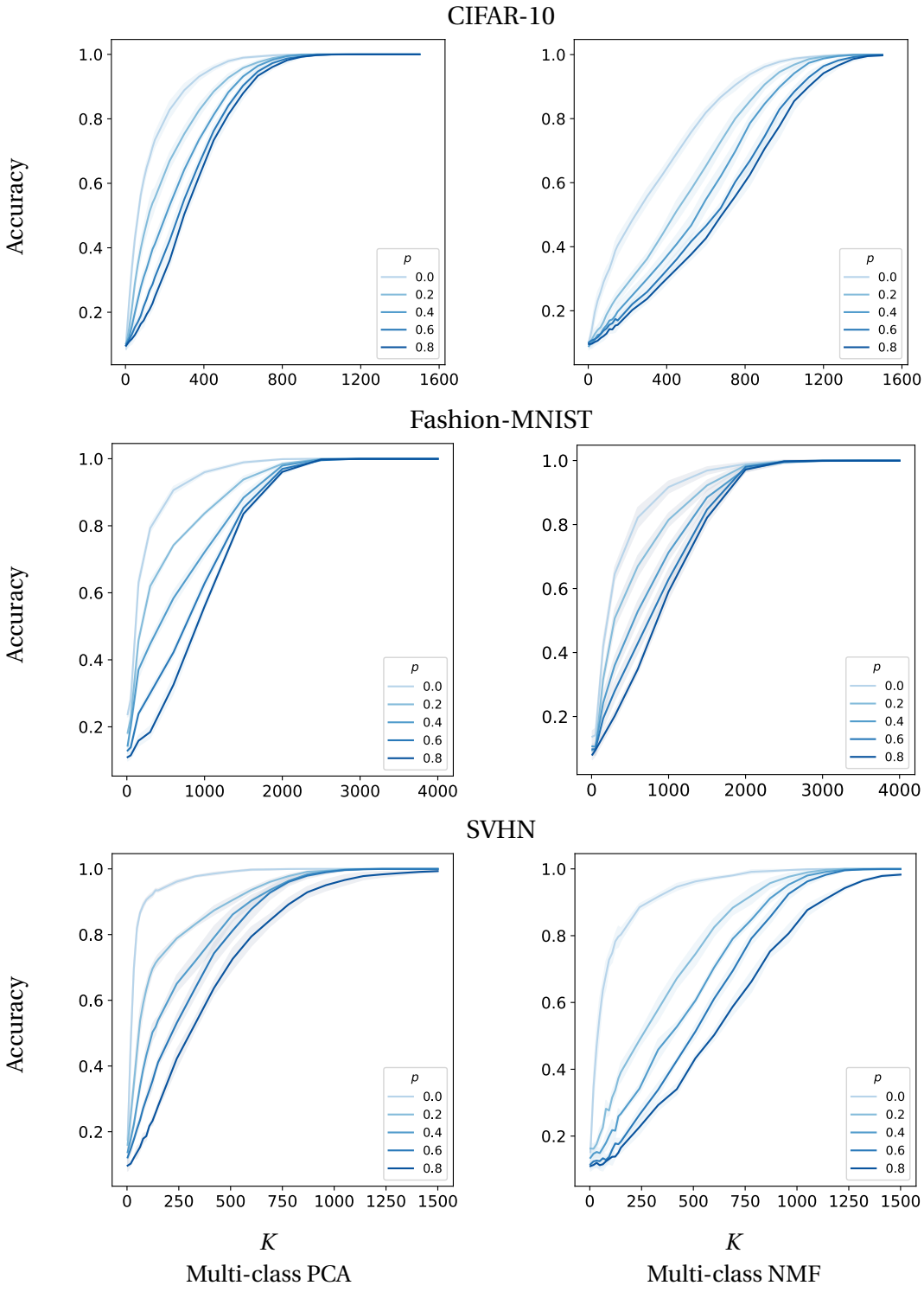


Figure 3.7 – Compared with Figures (3.5 and (3.6), using multi-class batches results in decreased sensitivity to memorization, as discussed in Section 3.4.1.

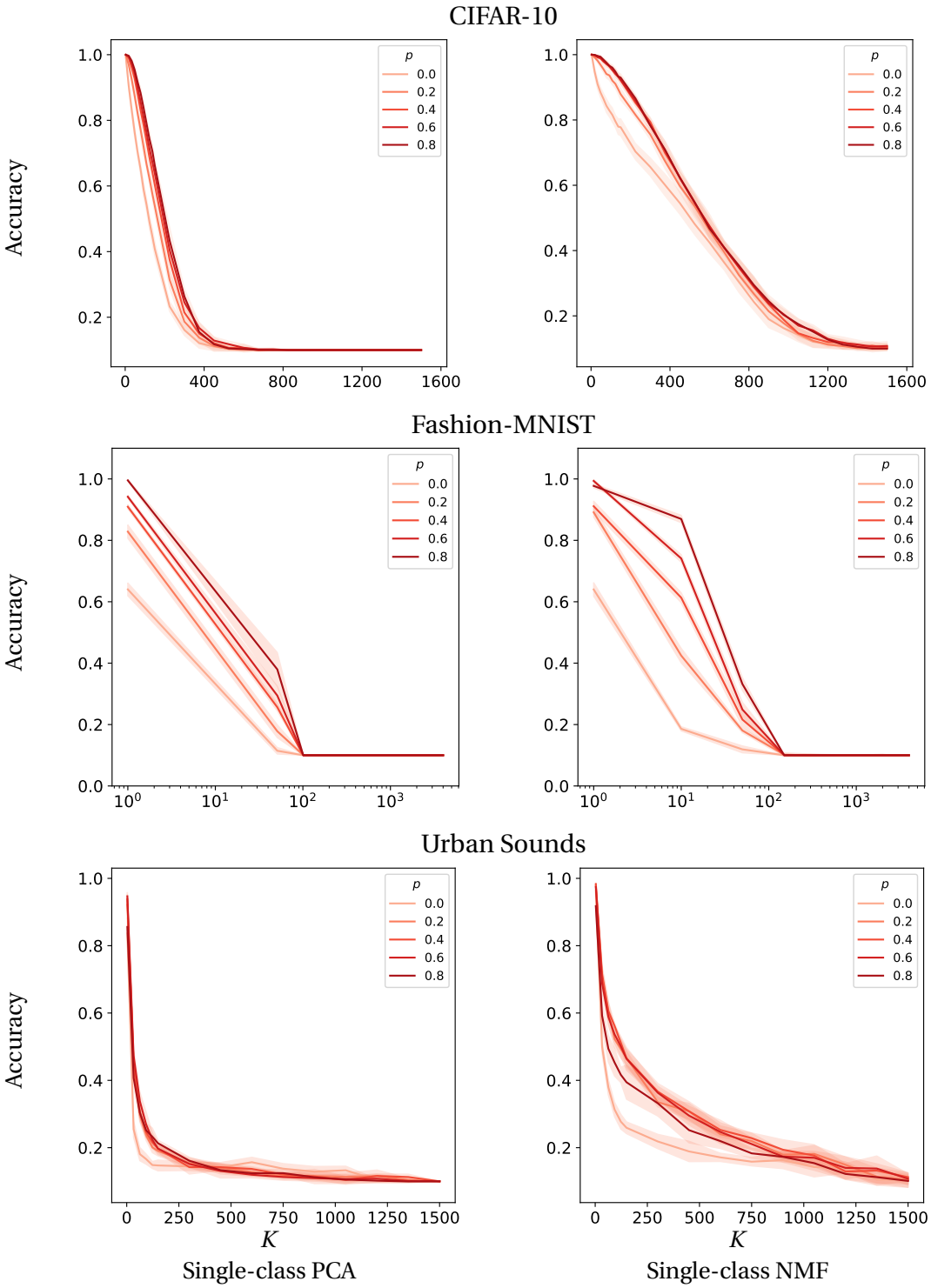
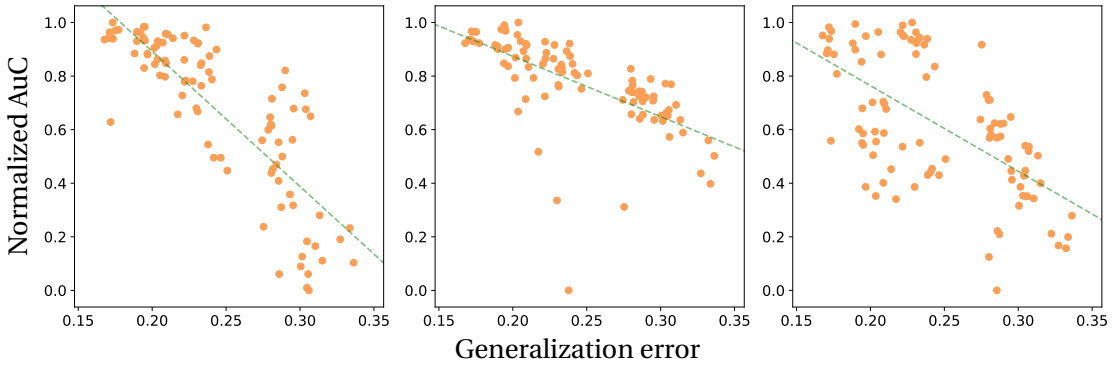


Figure 3.8 – NMF and PCA directions are more important in networks which do not memorize. Compared to Figures 3.5 and 3.6, where non-memorizing networks are more robust to random ablation (RA), in all cases we see they are less robust to ablation of NMF and PCA directions compared to memorizing networks.



(a) Single-class NMF (b) Single-class PCA (c) RA

Figure 3.9 – Area under the curve (AuC) of K-vs.-accuracy curves derived with NMF, PCA and RA. While all three methods show correlation with generalization error, NMF is most correlated with a Pearson correlation of -0.82, followed by PCA with -0.64 and RA with -0.61.

NMF thus gives us a tool to evaluate a network's *test* error, although it is based on *training* data. We test the usefulness of NMF as a proxy to the test error in the next section, by using NMF to determine when to perform early stopping.

Early Stopping

One feature of overfitting is that it does not happen all at once. In early training iterations, the test (or validation) error decreases along with training error. It is only later that the network starts to depend on noisy variations in the input to decrease the training error at the expense of the test error. Early stopping refers to training only up to the point when overfitting starts, and not past it.

While this is typically achieved using a validation set, in this experiment we test whether NMF can serve as an early stopping indicator while seeing only training data.

Once more, we trained CNNs on CIFAR-10 with the original labels, i.e., $p = 0$. Each network was trained for 10K batches with a batch size of 100. We recorded the test set error every 250 batches, and applied factorization to the deepest three convolutional layers using single-class NMF with a coarse grid on K. As before, we compute the area under each K vs. accuracy curve. Finally, we also computed the area under the curve produced by RA.

Results of two instances are shown in Figure 3.10 (a) and (b). We smooth the plots using a radius of two epochs to reduce noisy fluctuations. The matching-color dashed lines mark the local minima of the test loss in as well as the location of the first local maxima of the NMF and RA AuC curves after smoothing has been applied. We notice that the test loss minima align

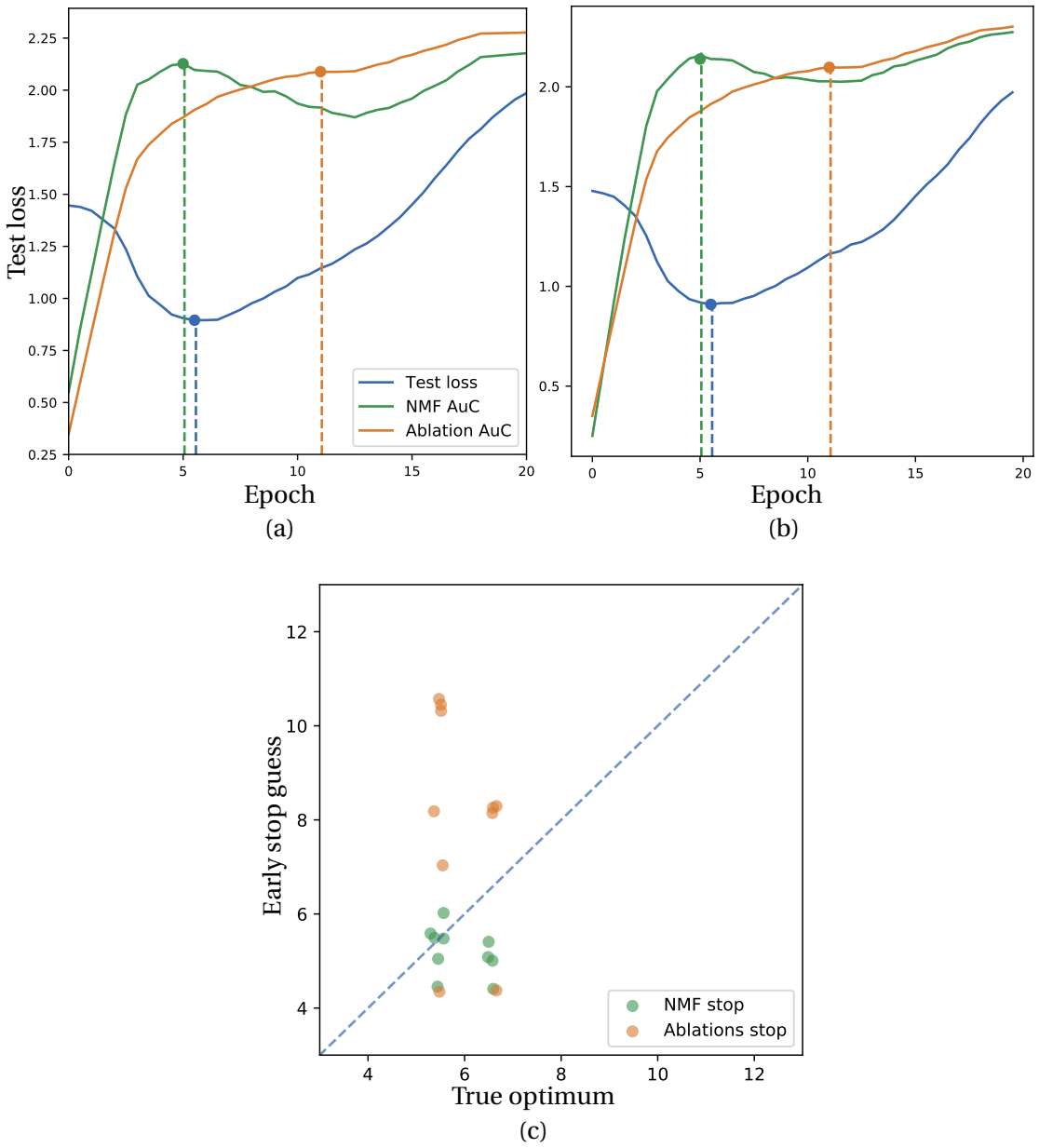


Figure 3.10 – Early stopping for CNN training on CIFAR-10. (a, b) The test loss is (in blue) starts to increase after about the 5th epochs, indicating the start of overfitting. Using single-class NMF, we can detect the test loss turning point. We show the area under the curve (AuC) computed with single-class NMF (in green), as discussed in section 3.5.2. Similarly, we show the AuC for RA (in orange). (c) The NMF AuC curve and test loss curve consistently have near extrema, as seen over several runs.

almost precisely with the maximum NMF AuC. We show more examples of this behavior in Figure 3.10 (c), where we compare the stopping times of NMF and RA against the best test loss over 10 different runs.

3.5.4 Experiments on VGG-19 and ImageNet

We perform a case study of VGG-19 [87], trained on ImageNet [80], since it is known for its good generalization ability and usefulness as a general feature extractor.

We apply NMF compression to the three deepest convolutional layers, for both single-class batches and multi-class batches. We selected 50 random classes from ImageNet and gathered batches of 50 training samples from each class.

In Figure 3.11 (a), accuracy of single-class batches under NMF (blue curve) exhibits a denoising effect which improves over the baseline top-1 accuracy (dashed line). As the constraint on K is relaxed, that accuracy drops back to the baseline level. This is in contrast to multi-class batches (green curve), where we regain baseline accuracy only when K is large. In Figure 3.11 (b) we see extreme sensitivity to single-class NMF ablations. Ablating multi-class NMF directions, however, has an impact similar to (d) ablating random axis-aligned directions.

In Figure 3.11 (c) we show a significant *per-class* correlation (Pearson $r = 0.78$) between NMF AuC and *test* accuracy as measured on batches from the ImageNet test set.

In the next chapter we will revisit VGG-19, and delve into the interesting properties of the NMF factorization $\boldsymbol{A} \approx \boldsymbol{UV}$ itself.

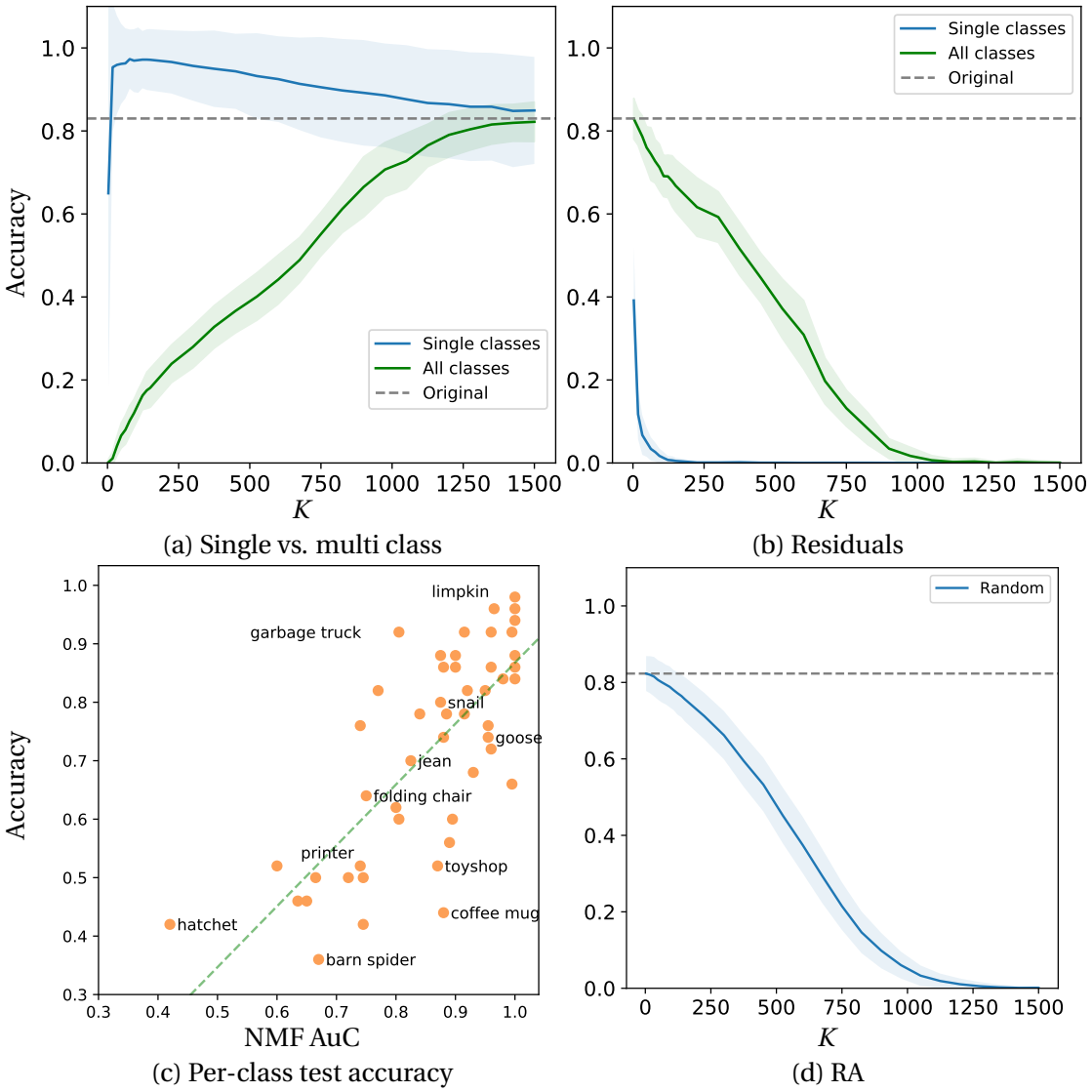
3.6 Conclusion

In this chapter we showed the relationship between compression through matrix factorization and memorization both in theory and in practice.

By extending a probabilistic view of k-means to NMF, we derived a bound over a the mutual information between specific input samples and their hidden activations. We proposed another view of NMF as a measure of the non-linearity of ReLU activation matrices.

Extensive empirical experiments confirmed that, indeed, NMF compression is effective at distinguishing between NNs of different levels of memorization and generalization. Our experiments suggest that this holds additionally during training and even per-class.

We conclude this chapter with a brief discussion of the computational overhead associated with NMF. Applying NMF compression to large matrices naturally incurs certain overhead. Our



(a) Single vs. multi class

(b) Residuals

(c) Per-class test accuracy

(d) RA

Figure 3.11 – (a) Single-class batches are highly compressible in deep VGG layers, as indicated by high accuracy for small values of K. Compression has a denoising effect, improving upon the baseline accuracy of the batch (dashed line), dropping back as K grows and capture more variation in the input. (b) Ablating single-class NMF directions causes a dramatic drop in accuracy. Ablating multi-class NMF directions, however, has an impact similar to (d) ablating random axis-aligned directions. (c) Per-class *test set* accuracy is significantly correlated with the area under the NMF K-vs.-accuracy curve (NMF AuC).

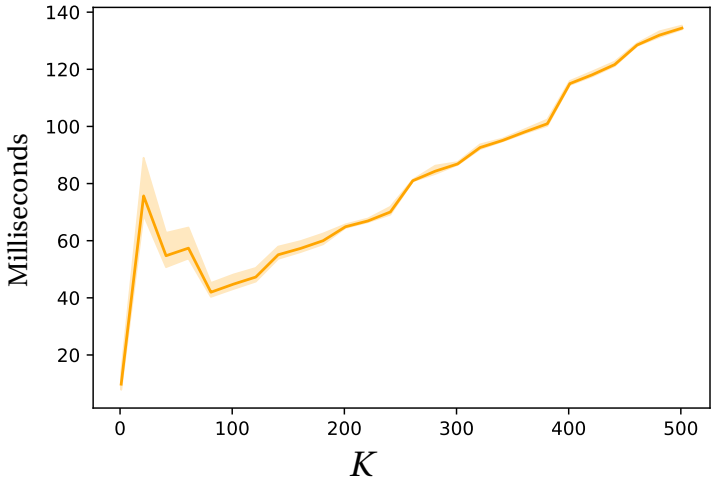


Figure 3.12 – NMF runtime on a typical ImageNet batch. Thanks to GPU acceleration, NMF with multiplicative updates can be run to convergence in reasonable time.

implementation of the NMF multiplicative update algorithm [60], however, runs in reasonable time thanks to GPU acceleration.

A typical batch used for VGG-19, i.e., 100 samples of 224×224 color images, is transformed to $100 \times 14 \times 14$ by layer `conv5_4`. The tensor flattens into a matrix of size 19600×512. In Figure 3.12 we show the timing curve for this batch as we increase K, using an NVIDIA Titan X. As can be seen, at $K = 500$ processing of the batch to convergence requires 197 milliseconds on average.

For a batch of $32x32$ CIFAR-10 images, where the deep feature maps are, say, 8×8, the batch processing time drop to approximately 135 milliseconds for $K = 500$. Sweeping over *all* values of K with an interval of 20 therefore takes about 2 seconds.

4 Semantic localization with matrix U

4.1 Introduction

In the previous chapter we established a relationship between non-negative matrix factorization applied to ReLU activations and memorization. More specifically, we showed that varying the factorization rank K affects classification performance in a way that is indicative of the level of memorization when compared across networks.

Recall that each application of NMF decomposes the activations as $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{V}$. In this and the next chapter, we delve into the individual components $\boldsymbol{U}$ and $\boldsymbol{V}$, and show they decompose the semantics learned by the CNN in a useful and revealing way.

In this chapter we focus on the matrix $\boldsymbol{U}$, which can be roughly interpreted as answering encoding "*where*". In other words, the matrix $\boldsymbol{U}$ holds soft clustering assignments for every patch given its representation in CNN feature space. The result is an unsupervised decomposition into semantic parts.

As neural networks become ubiquitous, there is an increasing need to understand and interpret their learned representations [65; 76]. In Section 4.2 the matrix $\boldsymbol{U}$ is shown to provide an interpretable window into how a CNN encodes objects in its hidden layer, by means of heatmaps showing which objects are considered similar and which are not.

Previous methods have been developed to explain CNN activations in terms of heatmaps [103; 83] (see Section 2.2.4). In these methods, heatmaps are derived by weighting the importance of each feature maps with respect to a particular output unit. These methods can therefore be

Some of the work presented in this chapter first appeared in [22].

seen as supervised, since the resulting heatmaps are associated with a designated output unit, which corresponds to an object class from a predefined set.

With NMF, however, heatmaps are *not* associated with an output unit or object class. Instead, they capture salient and common activation patterns in the input, as indicated by clusters in a deep CNN layer. This enables us to localize objects never seen before by the CNN, and for which there is no relevant output unit.

We evaluate NMF-based heatmaps on several tasks with subtle but important differences in naming:

- **Segmentation vs. Localization** is the difference between predicting pixel-wise binary masks and predicting bounding boxes, respectively.

- **Segmentation vs. co-segmentation** is the distinction between segmenting a single image into regions and jointly segmenting multiple images, thereby producing a correspondence between regions in different images (e.g., *cat*s in all images belong to the same segment).

- **Object co-segmentation vs. Part co-segmentation**. Given a set of images representing a common object, the former performs binary background-foreground separation where the foreground segment encompasses the entirety of the common object (e.g., *cat*). The latter, however, produces K segments, each corresponding to a *part* of the common object (e.g., *cat head*, *cat legs*, etc.).

In Section 4.3 we use NMF to perform co-segmentation of objects not in the original training set, such as *pyramid* (Figure 1.2), or object-parts such as the *head* or *torso* of an animal (Figure 4.6), which emerge in spite of training the CNN only with image level labels.

We find that parts form a *hierarchy* in feature space, e.g., the activations cluster for the concept *body*, which contains a sub-cluster for *limbs*, which in turn breaks down to *arms* and *legs* (see Figure 4.7).

In Section 4.4 we further evaluate NMF heatmaps by performing co-localization on a challenging real-world dataset, with significant clutter and object variation.

We validate our approach using several datasets and pre-trained CNNs, showing good performance across a variety of settings. In fact, in spite of using a pre-trained CNN with no fine tuning, co-localization with NMF achieves results comparable with the state-of-the-art.

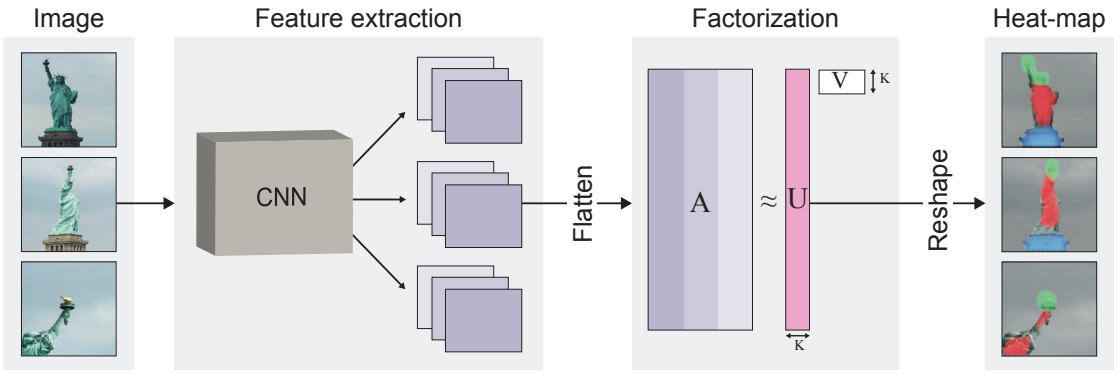


Figure 4.1 – An illustration of the NMF heatmap extraction pipeline. We obtain features from a deep CNN and view them as a matrix. We apply NMF to the feature matrix and reshape the resulting K factors into K heatmaps. See section 4.2 for a detailed explanation. Shown: VGG-19 heatmaps with $K = 3$ on the *Statue of Liberty* subset from iCoseg with.

4.2 NMF Heatmaps

We review in detail the procedure by which the matrix $\boldsymbol{U}$ is obtained and its relation to the input that generated it. Figure (4.1) gives an overview of the heatmap extraction pipeline.

4.2.1 CNN Feature maps

As before let an input image be a tensor of dimension $\mathbf{I} \in \mathbb{R}^{C_\mathbf{I} \times H_\mathbf{I} \times W_\mathbf{I}}$, where the final two dimensions are the height and the width of the image, respectively, and the third dimension is the number of channels, e.g., 3 for RGB images. The image $\mathbf{I}$ defines spatial grid, with the first dimension being a $C_\mathbf{I}$-dimensional feature representation of a particular spatial position. For an RGB image, this feature corresponds to color.

As the image gets processed layer by layer, the hidden activation at the ith layer of the CNN is the tensor $\mathbf{A}_i \in \mathbb{R}^{C_{\mathbf{A}_i} \times H_{\mathbf{A}_i} \times W_{\mathbf{A}_i}}$. For ease of notation, we drop the subscript i and $\mathbf{A}_i$ if there is no ambiguity. In most cases $H < H_\mathbf{I}$, $W < W_\mathbf{I}$ due to pooling operations commonly used in CNN pipelines. The number of channels C is user-defined as part of the network architecture, and in deep layers is usually between 256 and 1024.

Like the original image $\mathbf{I}$, the tensor $\mathbf{A}$ has a spatial interpretation as a *feature map*. The final two dimensions represent a spatial grid, where each position now corresponds to a *patch* of pixels in $\mathbf{I}$, and the first dimension forms a $C_\mathbf{A}$-dimensional representation of the patch.

A feature map represents multiple patches (depending on the size of image $\mathbf{I}$), and we view each one as a point inhabiting a single C-dimensional space, which we refer to as the CNN *feature space*.

4.2.2 NMF on feature maps

Feature space in deep layers is known to encode a high degree of semantic information, which can be distilled by applying NMF to a matrix of samples from that space.

As before, to apply matrix factorization we partially flatten $\mathbf{A}$ into a matrix:

$$\boldsymbol{A} \in \mathbb{R}^{(H \cdot W) \times C} \tag{4.1}$$

Note that the matrix $\boldsymbol{A}$ is effectively a "*bag of features*" in the sense that the spatial arrangement has been lost. The rows of $\boldsymbol{A}^i_{\mathcal{I}}$ can be permuted without affecting the result of factorization. We can naturally extend factorization to a set of N images, by vertically concatenating their patch features together:

$$\boldsymbol{A} = \begin{bmatrix} A^{(I_1)} \\ \vdots \\ A_{(I_N)} \end{bmatrix} \in \mathbb{R}^{P \times C} \tag{4.2}$$

where we use the superscript to indicate the image for which activations were produced and $P = \sum_i^N H_{\mathbf{A}^{(i)}} \cdot H_{\mathbf{A}^{(i)}}$. For ease of notation, we assume all images are of the same size, in which case $P = (N \cdot H \cdot W)$.

Having obtained $\boldsymbol{A}$ we proceed to factorize the matrix with a predefined rank K. In addition to NMF, we consider PCA as well.

After factorization we obtain $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{V}$. The kth factor ($1 \leq k \leq K$) is represented by the pair $(\boldsymbol{U}_{\cdot k}, \boldsymbol{V}_{k\cdot})$.

While $\boldsymbol{V}_{k\cdot}$ is a C-dimensional vector with some meaning in the CNN feature space, the matrix $\boldsymbol{U}$, of size $(N \cdot H \cdot W) \times K$, has as many rows as the activation matrix $\boldsymbol{A}$, one corresponding to every patch in every image.

Consequently, a single column $\boldsymbol{U}_k$ ($1 \leq k \leq k$) can be reshaped into $N \times H \times W$, and be viewed as a set of N **heatmaps**, each of dimension $H \times W$. These spatial dimensions are those of $\mathbf{A}_i$, and as such are often subsampled compared to the input images, e.g., by a factor of 16. We match the size of the heatmaps with that of the input images using bilinear interpolation. The N heatmaps can now be overlaid on top of their respective input images.

Repeating this procedure for each column of $\boldsymbol{U}$ creates K sets of N heatmaps, i.e., the tensor $\mathbf{U} \in \mathbb{R}_+^{N \times K \times H_\mathbf{I} \times W_\mathbf{I}}$. By giving each of the K sets a different color, we can overlay $\mathbf{U}$ with the original images all at once, as shown in Figure 1.2.

We note here that in early experiments with NMF we observed that some of the resulting factors would localize most strongly along the borders of the image. This predominantly occurred when batch normalization [46] was applied after zero-padded convolutions, as in ResNet-50. This combination amplifies edge artifacts present around the border. To resolve this problem and obtain clean heatmaps even in this case, we replaced all zero-padding with reflection-padding.

4.2.3 PCA heatmaps

It is interesting to compare the heatmaps generated by NMF to those generated by PCA. In Figures 4.2-4.5 we show NMF and PCA heatmaps for four categories from ImageNet. Since PCA coefficients can be negative, we do not overlay multiple heatmaps and instead show each of the $K = 3$ sets of maps in a different row.

A close inspection of PCA heatmap reveals they too hold a degree of semantics. Echoing back to the difficulty of interpreting PCA factors in pixel space as in Figure 2.13, in this case too it is not obvious how to handle the negative values. Flipping the sign of a column-row pair in the PCA $\boldsymbol{U}$ and $\boldsymbol{V}$ matrices, respectively, is also a solution to the PCA objective, and so the sign of any given solution is arbitrary.

While some processing heuristics might be applied, such as thresholding each PCA map into two positive maps, we do not pursue that direction in this thesis.

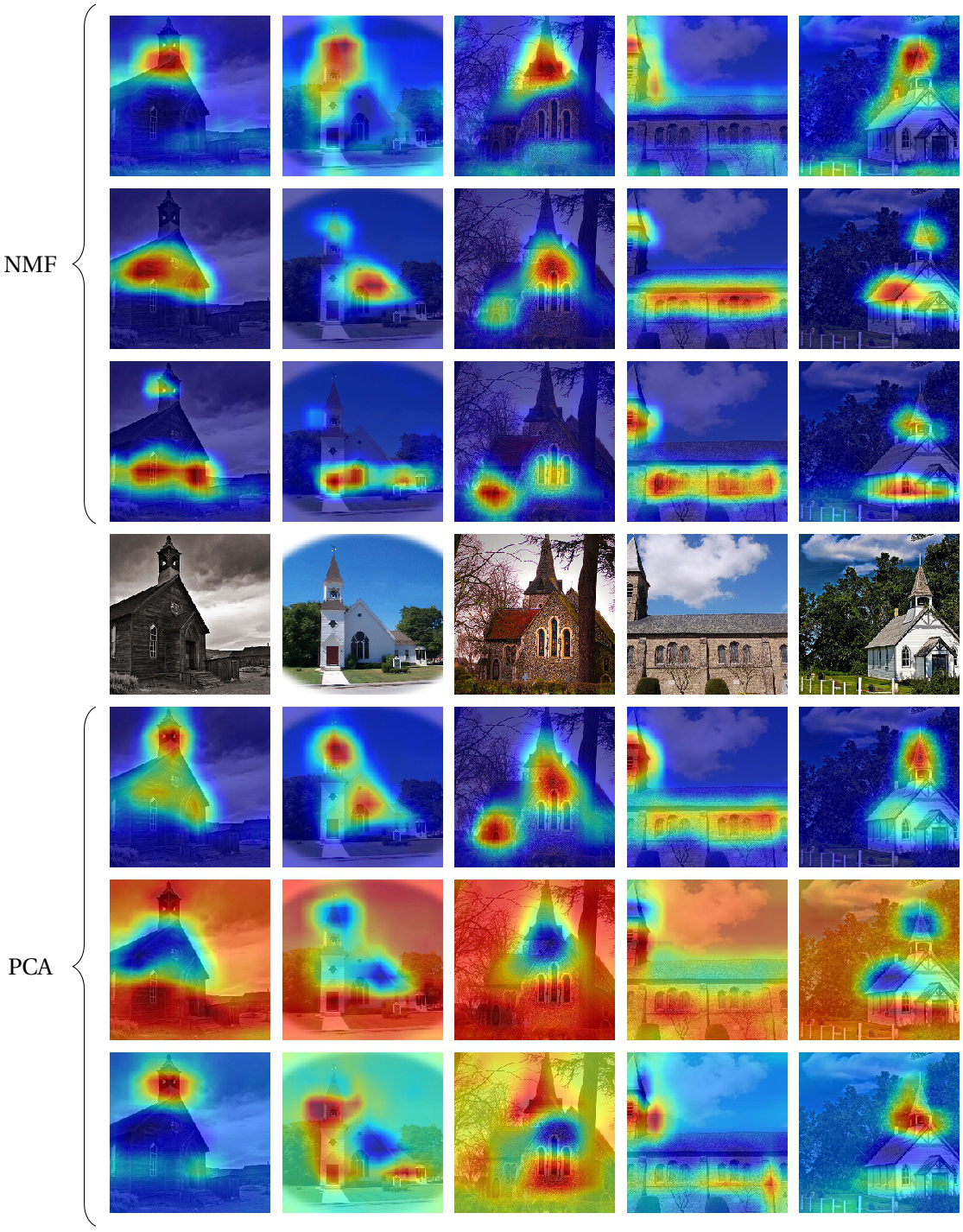


Figure 4.2 – An example of NMF and PCA heatmaps with $K = 3$ (one factors per row) derived from VGG-19 `conv5_4`. While NMF factors can be directly interpreted as saliency maps, principal components are less straightforward to interpret, and require additional post-processing. Here shown are images from ImageNet class 497, *church building*.

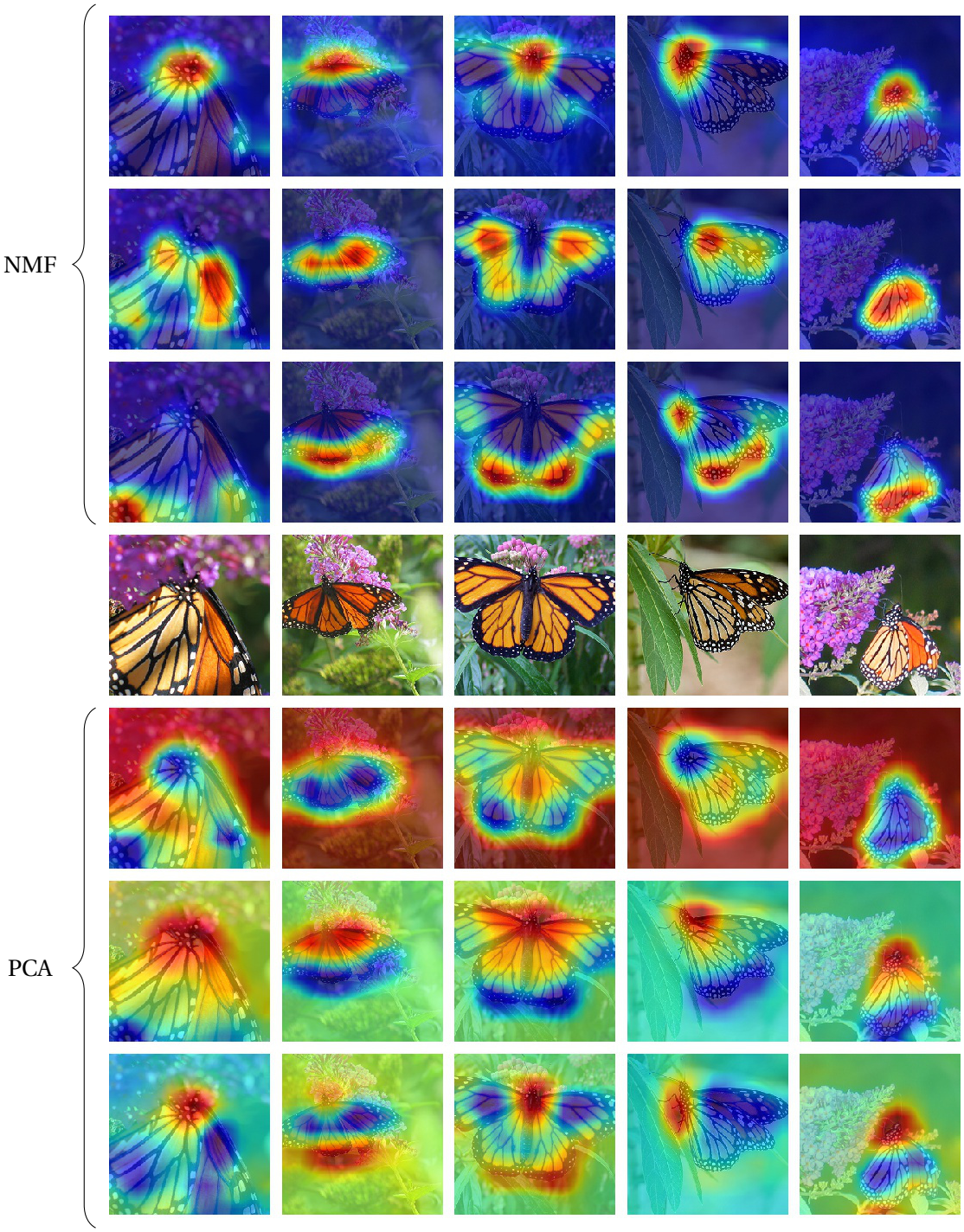


Figure 4.3 – An example of NMF and PCA heatmaps with $K = 3$ (one factors per row) derived from VGG-19 `conv5_4`. While NMF factors can be directly interpreted as saliency maps, principal components are less straightforward to interpret, and require additional post-processing. Here shown are images from ImageNet class 323, *monarch butterfly*.

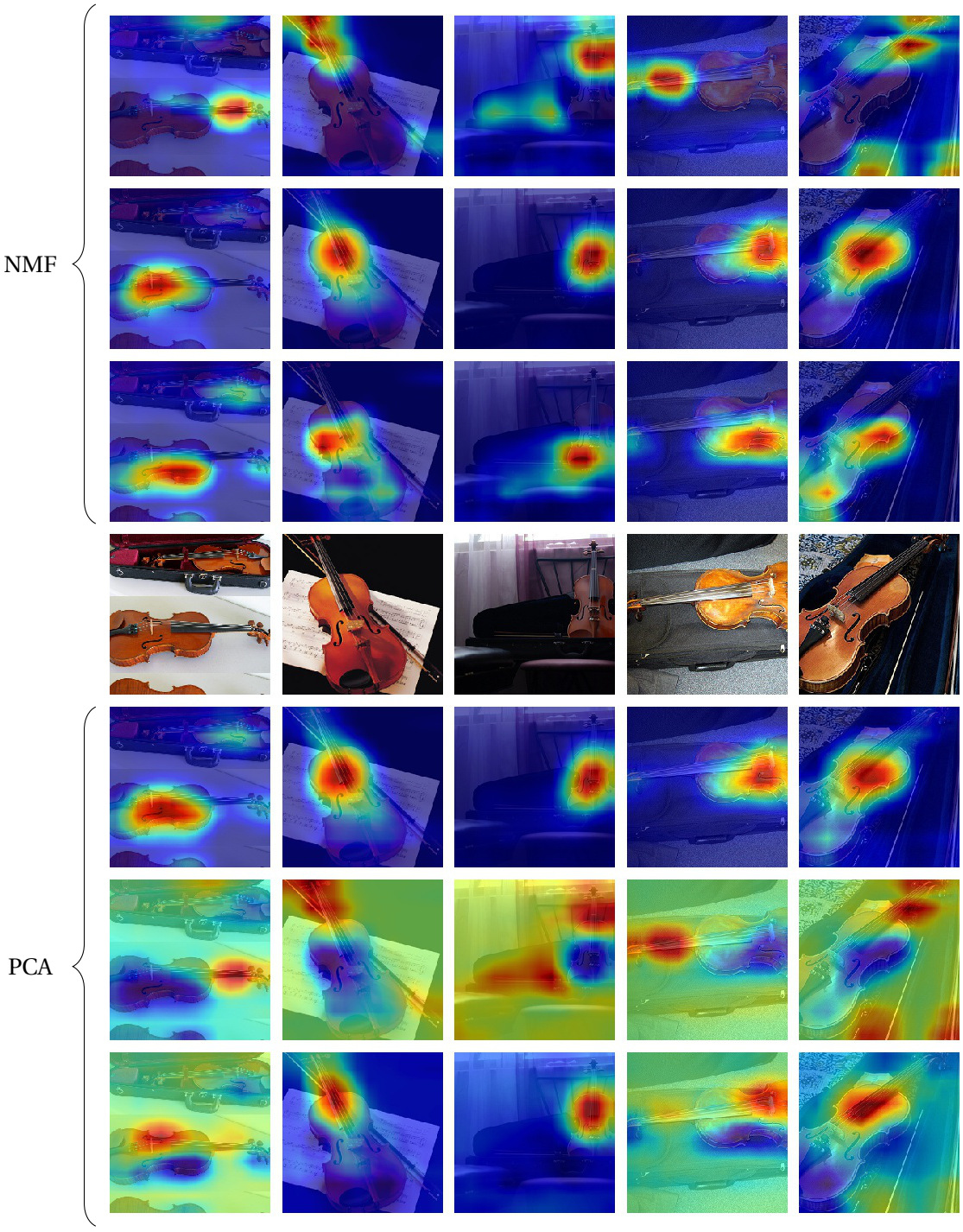


Figure 4.4 – An example of NMF and PCA heatmaps with $K = 3$ (one factors per row) derived from VGG-19 `conv5_4`. While NMF factors can be directly interpreted as saliency maps, principal components are less straightforward to interpret, and require additional post-processing. Here shown are images from ImageNet class 889, *violin, fiddle*.

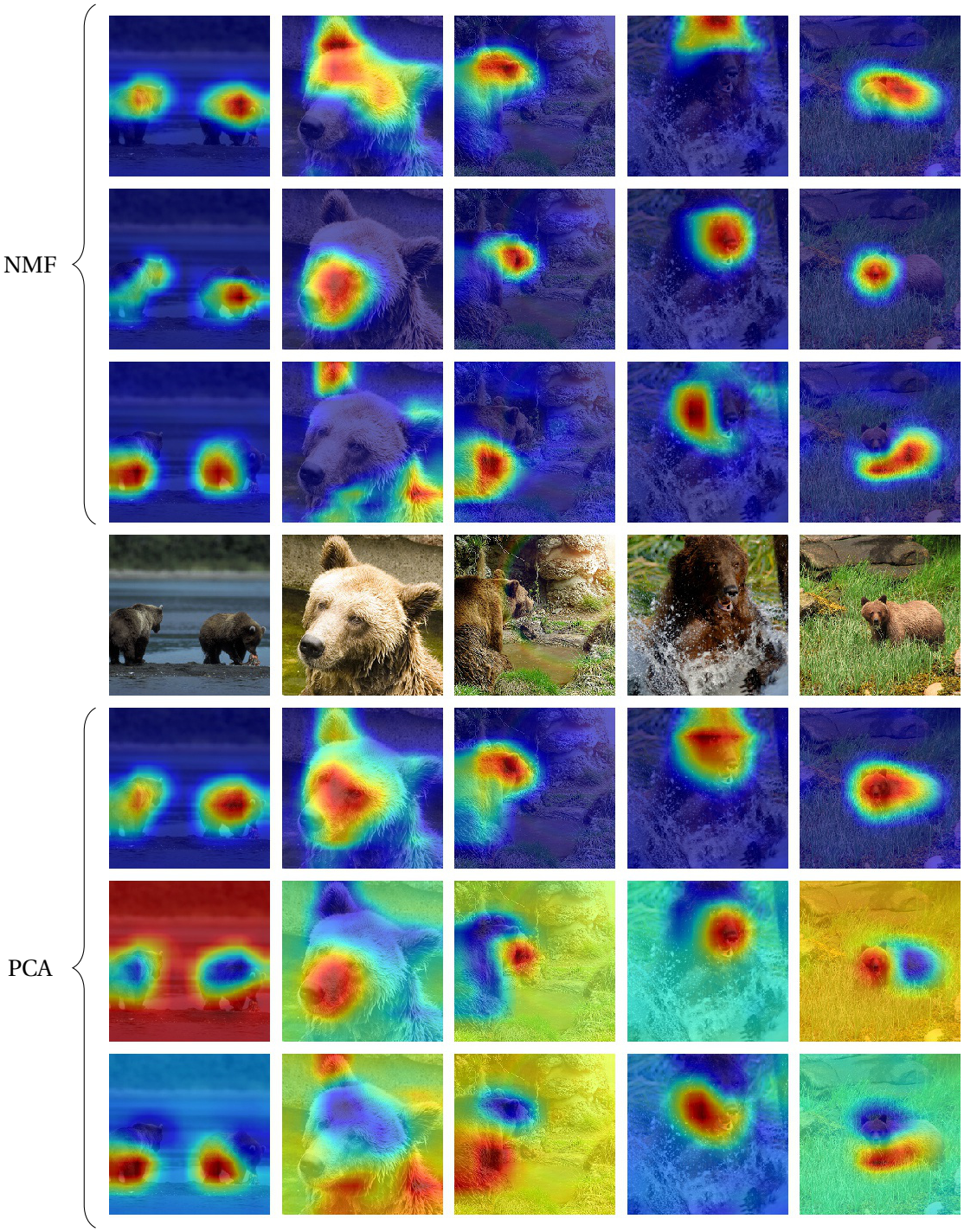


Figure 4.5 – An example of NMF and PCA heatmaps with $K = 3$ (one factors per row) derived from VGG-19 `conv5_4`. While NMF factors can be directly interpreted as saliency maps, principal components are less straightforward to interpret, and require additional post-processing. Here shown are images from ImageNet class 294, *brown bear*.

4.3 Experiments on iCoseg

The iCoseg dataset [9] is a popular benchmark for co-segmentation methods. It consists of 38 image sets, where each image is annotated with a pixel-wise mask encompassing the main object common to the set. Images within a set are uniform in that they were all taken on a single occasion, depicting the same object(s). The challenging aspect of this dataset lies in the significant variation of viewpoint, illumination, and object deformation.

We chose five sets and further labeled them with pixel-wise object-part masks, namely the categories *Elephants*, *Taj Mahal*, *Pyramid*, *Gymanstics1*, *Statue of Liberty*. This process involved splitting the given ground truth whole-object masks into individual parts. We also annotated common background objects, e.g., *animal* in the *Pyramids* set (see Figure 1.2). The number of images in these sets ranges from as few as 5 up to 41. When comparing against [94] and [78] in Table 4.2, we used the subset of iCoseg used in those papers.

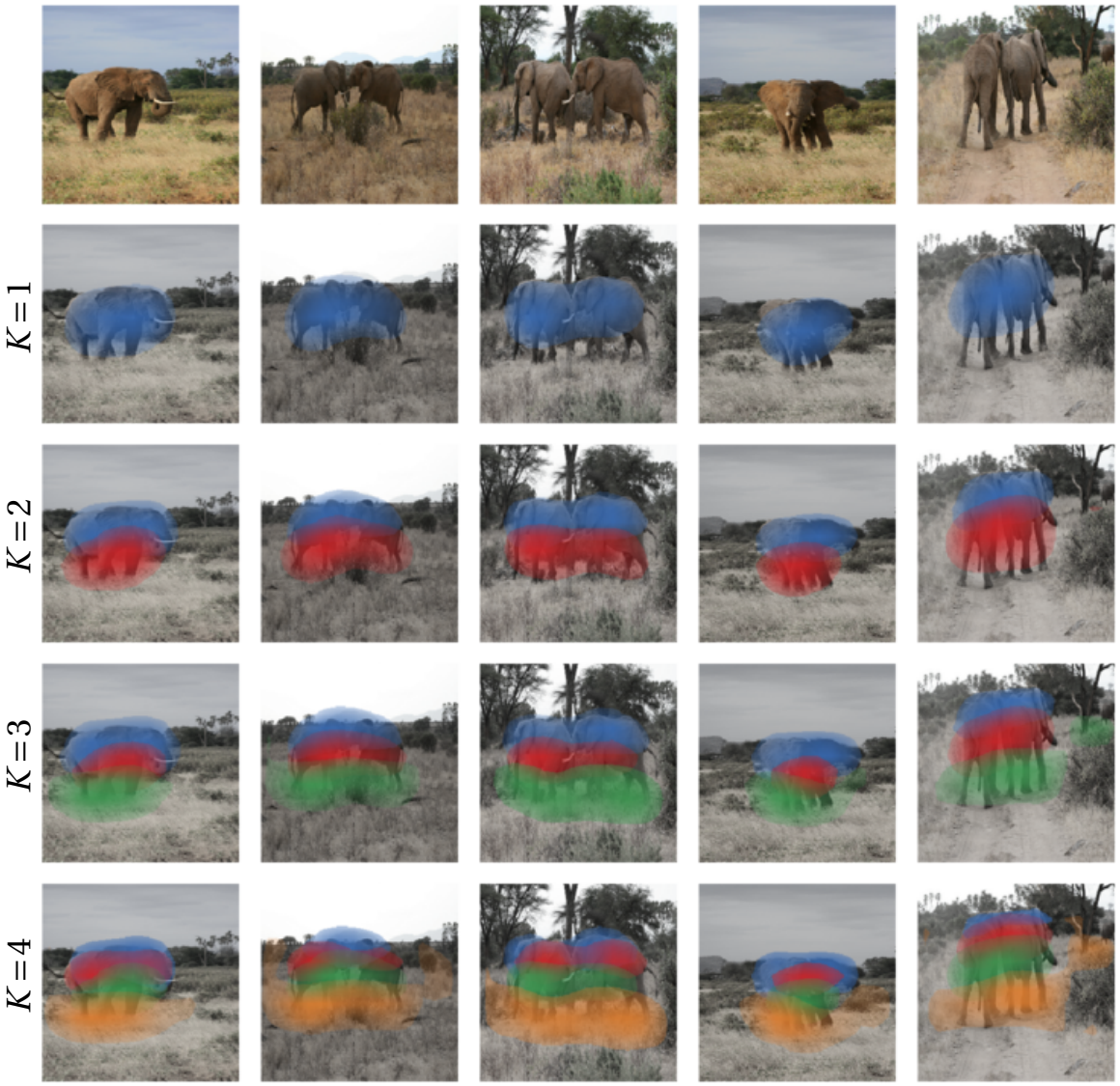
4.3.1 Qualitative investigation

For each set in iCoseg, we obtained activations from two CNNs. For VGG-19 we used the deepest convolutional layer, `conv5_4`, and for ResNet-50 we used the last layer of the third convolutional block. We then applied NMF to these activations with increasing values of K.

VGG-19 In Figures 4.6 and 4.7 we present VGG-19 results for two image sets, *Elephants* and *Gymanstics1*, respectively. We see a clear correspondence between NMF factors and coherent object-parts, however, the heatmaps are coarse. Due to the low resolution of deep CNN activations, and hence of the heatmap, we get blobs that do not perfectly align with the underlying region of interest.

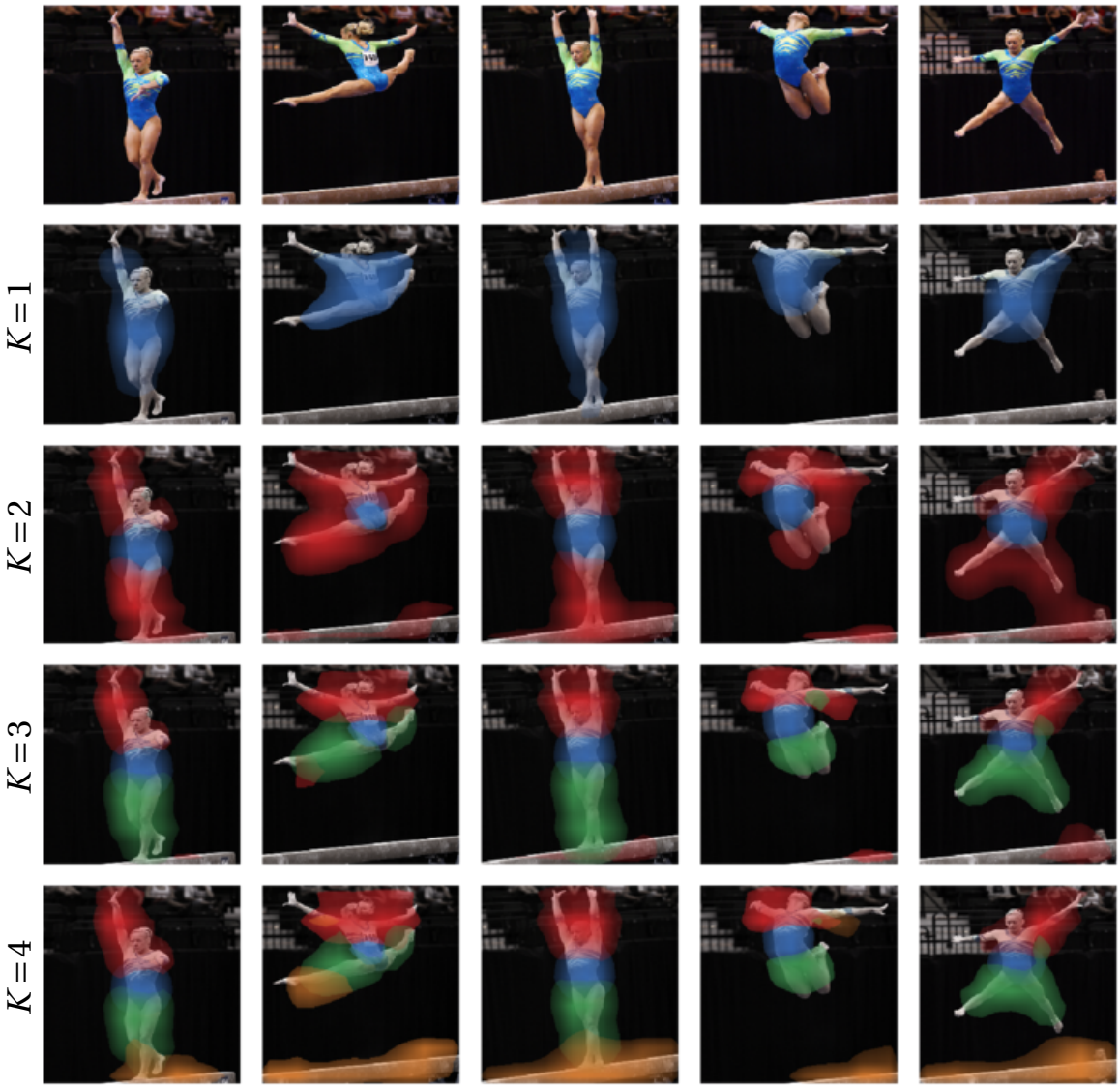
We notice that when $K = 1$, the single NMF factor corresponds to a whole object, encompassing multiple object-parts. This, however, is not guaranteed, since it is possible that for a set of images, setting $K = 1$ will highlight some *background* element rather than the foreground. Nonetheless, as we increase K, we get a decomposition of the object or scene into individual parts. This behavior reveals a hierarchical structure in the clusters formed in CNN feature space.

For instance, in Figure 4.7, we can see that $K = 1$ encompasses most of gymnast's body, $K = 2$ distinguished her midsection from her limbs, $K = 3$ adds a finer distinctions between arms and legs, and finally $K = 4$ adds a new component that localizes the beam. This observation also indicates the CNN has learned representation that 'explains' these concepts with invariance to pose, e.g., leg positions in the 2nd, 3rd, and 4th columns.



VGG-19 heatmaps for *Elephants*

Figure 4.6 – NMF with incremental K on the *Elephants* subset from iCoseg with VGG-19. Each row shows a separate factorization where only K is changed. Different colors correspond to the heatmaps of the K different factors. NMF factors correspond well to distinct object parts. This Figure visualizes the data in Table 4.1, where heatmap color corresponds with row color.



VGG-19 heatmaps for *Gymnastics1*

Figure 4.7 – NMF with incremental K on the *Gymnastics1* subset from iCoseg with VGG-19. Each row shows a separate factorization where only K is changed. Different colors correspond to the heatmaps of the K different factors. NMF factors correspond well to distinct object parts. This Figure visualizes the data in Table 4.1, where heatmap color corresponds with row color.

A similar decomposition into legs, torso, back, and head can be seen for the elephants in Figure 4.6. This shows that we can localize different objects and parts even when they are all common across the image set.

Interestingly, the decompositions shown in the introductory Figure 1.2 exhibit similar high semantic quality in spite of their dissimilarity to the ImageNet training data, as neither pyramids nor the Taj Mahal are included as class labels in that dataset.

We also note that as some of the given sets contain as few as 5 images (Figure 1.2b comprises the whole set), our method does not require many images to find meaningful factors.

ResNet-50 In Figures 4.8 and 4.9 we show similar heatmaps, extracted using ResNet-50. Compared to VGG-19 heatmaps, these heatmaps are considerably more *dense*.

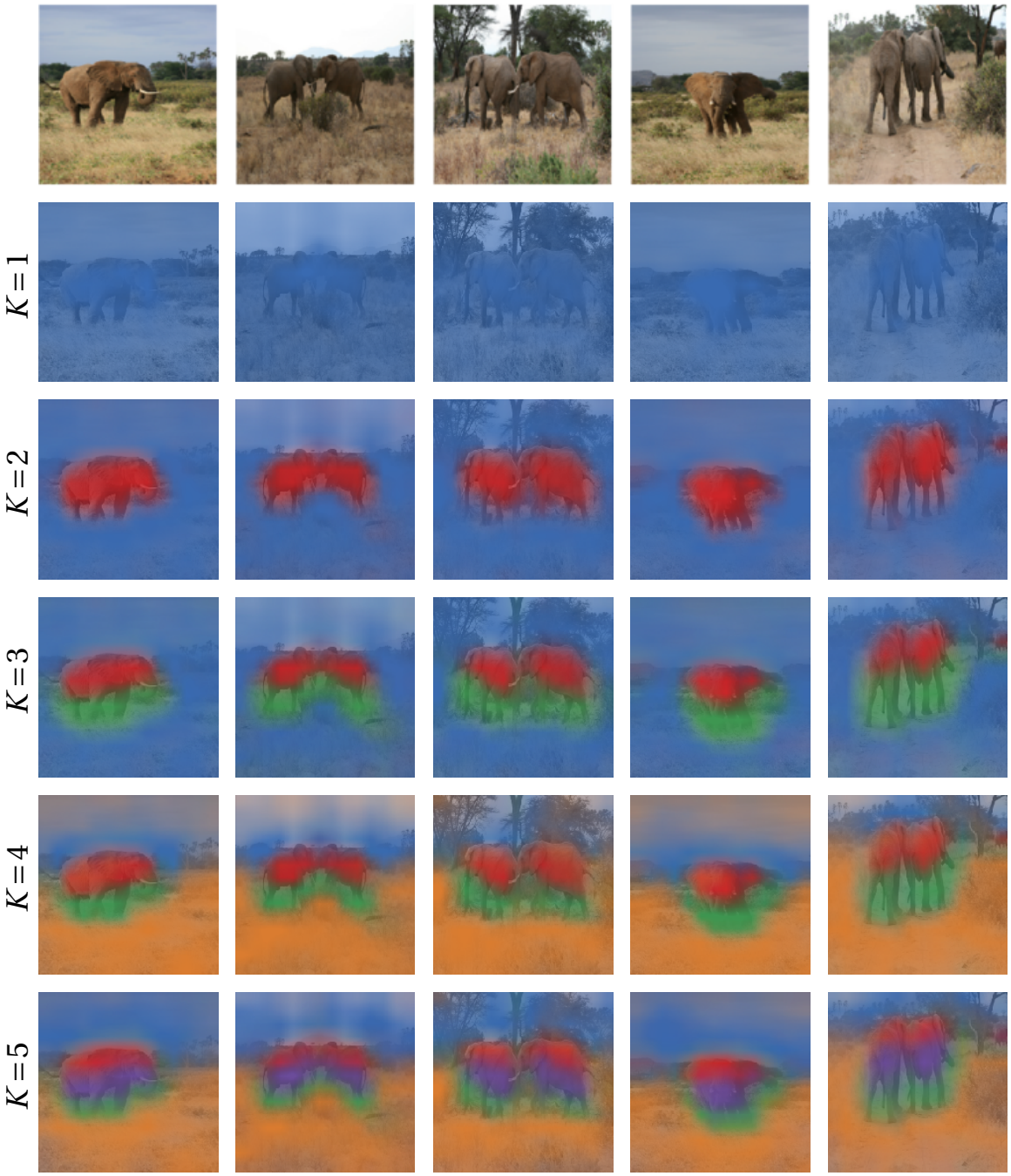
To explain this difference be examined the VGG-19 and ResNet-50 feature maps themselves. A likely cause for this reason is the fact that VGG-19 and VGG-16 deep activations are 92% sparse on iCoseg, whereas with ResNet-50 activations are only 71% sparse for the same data.

The main differences between the two architectures is ResNet's use of batch-normalization and residual layers. We tested whether the former is the trigger for the denser activation by examining the activations of VGG-16 BN and VGG-19 BN, i.e., versions of the basic VGG architecture with added batch-normalization. In this case too, however, VGG-16 BN and VGG-19 BN activation proved about 90% sparse. We conclude therefore that the difference is a characteristic of the residual architecture itself.

With $K = 1$, ResNet-50 produces a heatmap which encompasses the whole image, as opposed to VGG-19 which preferred the salient object. In both Figures 4.8 and 4.9, $K = 2$ results in clear foreground-background separation, with one of the components singling out the salient object. This too, however, is not guaranteed, since setting $K = 2$ could result in a separation between background elements.

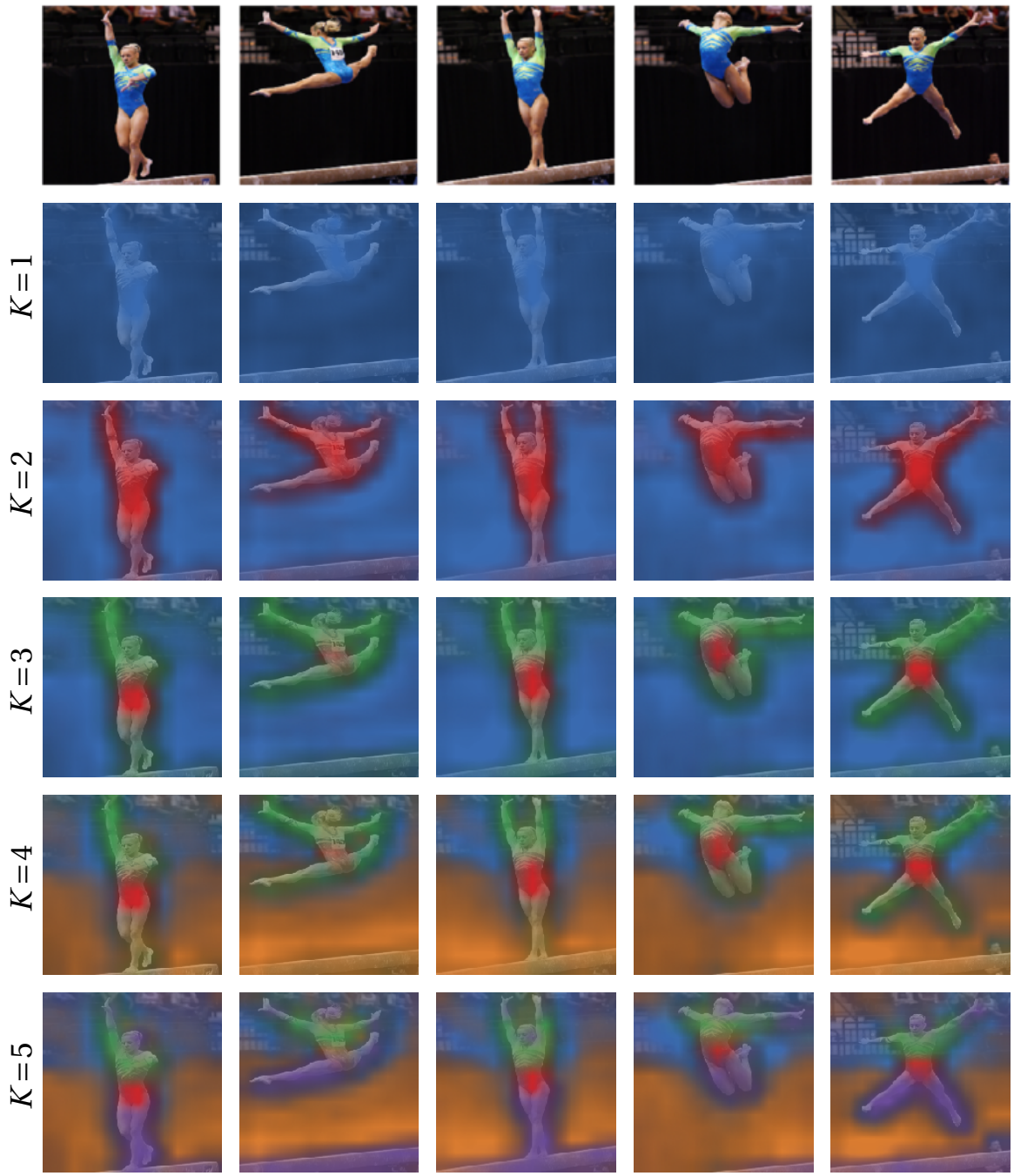
As we increase K, we find that factors are indeed allocated to describing variations in the background. For instance, in Figure 4.8, the *sky* and the *ground* are each assigned a factor. Increasing to, e.g., $K = 7$ (not shown) gives a factor that isolated the dark green trees in the background.

As a result it is only with relatively high factorization ranks, relative to VGG-19, that ResNet heatmaps contain parts of the same 'resolution' as VGG-19.



ResNet-50 heatmaps for *Elephants*

Figure 4.8 – NMF with incremental K on the *Elephants* subset from iCoseg with ResNet-50. Compared to Figure 4.6, ResNet produces heatmaps with considerably more dense background activation. Increasing K is as likely to distinguish between background elements (e.g. *ground* vs *sky* vs *tree*) as between parts of the foreground object.



ResNet-50 heatmaps for *Gymnastics1*

Figure 4.9 – NMF with incremental K on the *Gymnastics1* subset from iCoseg with ResNet-50. Compared to Figure 4.7, ResNet produces heatmaps with considerably more dense background activation. Increasing K is as likely to distinguish between background elements as between parts of the foreground object.

4.3.2 Object and part co-segmentation

Given the heatmaps $\mathbf{U} \in \mathbb{R}_+^{N\times K\times H\times W}$ and ground truth part segmentations masks $\mathbf{G}\{0,1\}^{N\times T\times H\times W}$, we would like to quantify the correspondence of NMF heatmaps to those parts. A high score indicating a match attests to the semantic meaning of the NMF factors.

Since NMF factors are unsupervised, a single factor may correspond not to a single part, but to a composition of several parts, e.g. *limbs*, as observed in the examples of the previous section. Conversely, a single part might correspond to a composition of several factors (e.g., the wheels in Figure 4.11c).

We must therefore associate groups of factors with groups of parts. This can be done with respect to two different objectives. The first objective is *exploratory*, meaning we have complete access to ground truth data. We can then estimate the semantics of a factor vs. part combination simply by evaluating all ground truth data.

This is the strategy used in [10], with the crucial difference being that they consider each individual CNN feature map, of which there are typically hundreds or even thousands, whereas we consider only K NMF heatmaps, where K is usually small. In addition, each NMF heatmap is a combination of all CNN heatmaps, which allows for correlations not present when considering individual feature maps.

The second objective is *predictive*, i.e., we produce an association between factors and parts using none or only a fraction of ground truth data, and evaluate the match on the remainder of the ground truth.

For instance, for object co-segmentation we need to produce a single heatmap encompassing the object of interest. Based on the examples shown in Figures 4.6 and 4.7, a viable strategy for VGG-19 is to simply set $K = 1$. This will produce a heatmap likely to surround the salient object.

Similarly, inspired by Figures 4.8 and 4.9, a strategy for ResNet-50 can be to set $K = 2$ and select the foreground factor. This can be done with a simple heuristic: if factor a has less activation on the 1-pixel-wide border of the heatmap than factor b, then factor a is the foreground.

Finally, a more robust solution is to use a small subset of ground truth, e.g., a single ground truth mask, to associate factors and parts. We can then test the quality of the match on the remaining images in the set.

Our matching and evaluation proceeds as follows. We first binarize the NMF heatmaps $\mathbf{U}$, producing $\mathbf{B}$. We then augment the collections of factor and part heatmaps by merging certain subsets. Finally, we measure the match between (sets of) factors and parts using the intersection-over-union (IoU) measure, also known as Jaccard similarity. We review these

steps below, describing the merging procedure last.

Extracting B Recall that the tensor U is reshaped from the NMF matrix $\boldsymbol{U}$. As such, its scale is arbitrary. We therefore divide each set $\mathsf{U}_{\cdot k \cdots}$, $1 \leq k \leq K$ by the maximal number in that set.

This simple heuristic approximates the probabilistic interpretation of NMF presented in the previous chapter, with the assumptions that there is at least one point per factor that is highly probable with respect to that factor, which we set to 1.

To binarize the normalized maps, we can use a simple threshold, e.g. $\mathsf{B} = \text{normalize}(\mathsf{U}) \geq \tau_{\text{binarize}}$. We experimentally set $\tau_{\text{binarize}} = 0.75$.

Measuring IoU We compute the IoU as follows:

$$\boldsymbol{B} = \textbf{flatten}(\mathsf{B}),\ \boldsymbol{G} = \textbf{flatten}(\mathsf{G}) \tag{4.3}$$

$$\text{IoU}_{k,t} = \frac{\boldsymbol{B}_k^\top \boldsymbol{G}_t}{\sum \boldsymbol{B}_k + \sum \boldsymbol{G}_t - \boldsymbol{B}_k^\top \boldsymbol{G}_t} \tag{4.4}$$

The IoU is also the probability of a pixel i being 1 in both masks, *given* that it is 1 in one of them. It therefore equals 1 when overlap between the masks is exact, and is 0 when they are mutually exclusive.

The computation above results in a $K \times T$ matrix of IoU scores. We consider a factor k and a part t as matching only when $\text{IoU}_{k,t} > \text{IoU}_{k',t} \wedge IoU_{k,t} > \text{IoU}_{k,t'}\ \forall\ 1 \leq k' \leq K,\ 1 \leq t' \leq T$.

Merging factors and parts We propose to look at the two quantities related to the IoU, and use them to augment B or G.

The first term, intersection-over-parts, is the probability of part t being entirely contained in factor k:

$$\text{IoF}_{k,t} = \frac{\boldsymbol{B}_k^\top \boldsymbol{G}_t}{\sum \boldsymbol{B}_k} \tag{4.5}$$

This information is used only to create new ground truth targets, and so the evaluated NMF heatmaps are independent of the ground truth data. In other words, this type of augmentation is possible even in the unsupervised case.

The second term, intersection-over-factors, is the probability of factor k being entirely contained in a part t:

$$\mathrm{IoP}_{k,t} = \frac{\boldsymbol{B}_k^{(n)\top} \boldsymbol{G}_t^{(1)}}{\sum \boldsymbol{G}_t^{(n)}} \tag{4.6}$$

This information is used to create *new* NMF heatmaps which are *not* independent of the ground truth data. This means that factor augmentation is not possible in the unsupervised case. In the partial data case, $n < N$ is the number of images whose ground truth contributes to augmentation, e.g. $n = 1$ for "one-shot" prediction, and finally $n = N$ in the fully exploratory setting.

Finally, let τ_{merge} be a set threshold, we augment the set of binary factors B by concatenating up to T composite factors:

$$\mathsf{B}'_t = \sum \left\{ \mathsf{B}_k | \mathrm{IoF}_{t,k} \geq \tau_{\text{merge}} \right\} \tag{4.7}$$

$$\mathsf{B} \leftarrow \begin{bmatrix} \mathsf{B} \\ \left\{ \mathsf{B}'_t \middle| |\mathsf{B}'_t| \geq 2 \right\} \end{bmatrix} \tag{4.8}$$

And similarly by concatenating up to K merged parts:

$$\mathsf{G}'_k = \sum \left\{ \mathsf{G}_t | \mathrm{IoP}_{t,k} \geq \tau_{\text{merge}} \right\} \tag{4.9}$$

$$\mathsf{G} \leftarrow \begin{bmatrix} \mathsf{G} \\ \left\{ \mathsf{G}'_k \middle| |\mathsf{G}'_k| \geq 2 \right\} \end{bmatrix} \tag{4.10}$$

In all our experiments we empirically set $\tau_{merge} = 0.2$. Given these augmented sets, we obtain IoU results as described above.

Results

For VGG-19, Table 4.1 shows the matching parts and IoU scores for factors extracted from the five image sets of iCoseg that we have annotated. We used all the ground truth to associate factors and parts. In this case, no factor merging was needed, but part merging is common.

These scores correspond to the visualizations of Figures 1.2, 4.6 and 4.7, confirming what we observe qualitatively. As can be seen, although the factors align well visually with their respective parts, the IoU can still low due to the low-resolution of the original heatmaps.

In Table 4.2 report results for object co-segmentation, i.e., $T = 1$ and the set G contains a single

ground truth 'part', corresponding to the whole object.

We use VGG-19 with the $K = 1$ heuristic and ResNet-50 with the $K = 2$ heuristic described above. We also considered "one-shot" co-segmentation, where we used a single ground truth instance to determine which factors belong in the foreground. In that case we used heatmaps from several application of NMF, with $K = \{2, 4, 6, 8\}$.

We include results of several state-of-the-art co-segmentation methods for comparison. The *supervised* method of Vicente et al. [94] chooses among multiple segmentation proposals per image by learning a regressor to predict, for pairs of images, the overlap between their proposals and the ground truth. Input to the regressor included per-image features, as well as pairwise features.

The methods Rubio et al. [78] and Rubinstein et al. [77] are unsupervised and rely on a Markov random field formulation, where the unary features are based on surface image features and various saliency heuristics. For pairwise terms, the former method uses a per-image segmentation into regions, followed by region-matching across images. The latter approach uses a dense pairwise correspondence term between images based on local image gradients.

These methods employ heavy use of *surface features*, e.g., edges and colors in pixel space, to obtain a pixel-accurate segmentation, and as a result, they achieve high intersection-over-union scores.

The objective of our experiments, however, is to asses the semantics of learned CNN features, not maximize the segmentation accuracy. As a result, we do not apply heavy post processing to our heatmaps. Nonetheless, in spite of being based on low-resolution heatmaps, NMF heatmaps compare favorably against these domain-specific methods, even outperforming them in some cases.

We can see that in most of these cases, the $K = 1$ and $K = 2$ heuristics work well for VGG-19 and ResNet-50, respectively, where the category *Pyramids* is an exception for both. Shown in Figure 1.2, the salient region in this image set does, in fact, include more than the eponymous object of interest. The one-shot methods overcome this issue since we essentially over-segment the image set by setting a large K, and use the single ground truth sample to aggregate the relevant segments. In all cases, however, ResNet-50 produces better results than any VGG network.

	Elephants		Taj Mahal		Pyramids		Gymnastics1		Statue of Liberty	
VGG-19, K=2	*torso/back/hea*	59	*dome*	33	*animal*	36	*torso/waist*	35	*torso*	36
	torso/leg	35	*tower/building*	46	*pyramid*	56	*arm/leg/head*	20	*torch/base/head*	28
VGG-19, K=3	*back/head*	46	*building*	45	*backgroun*	27	*torso/waist*	38	*base*	14
	torso	25	*dome*	40	*pyramid*	55	*arm/head*	22	*torso*	39
	leg	21	*tower*	13	*animal*	36	*leg*	33	*torch/head*	23
VGG-19 , K=4	*torso/back/hea*	58	*building*	72	*backgroun*	27	*torso/waist*	40	*torso*	39
	head	36	*dome*	43	*pyramid*	52	*torso/arm/hea*	33	*background*	44
	torso	20	*background*	08	*animal*	37	*leg*	37	*torch/head*	26
	leg	16	*tower*	16	*person*	12	*background*	14	*base*	40

Table 4.1 – Object and part co-segmentation on five iCoseg image sets using VGG-19. Part-labels are automatically assigned to NMF factors using all available ground truth, and are shown with their corresponding IoU-scores. These results show that clusters in CNN feature space correspond to coherent parts. More so, the results indicate the presence of a cluster hierarchy in CNN feature space, where part-clusters can be seen as sub-clusters within object-clusters (See Figures 1.2, 4.1, 4.6 and 4.7 for visual comparison. Cell color corresponds with heatmap color).

Method	Supervision	Elephants	Taj Mahal	Pyramids	Gymnastics1	Statue of Liberty
Vicente [94]	Supervised	43	91	-	-	94
Rubio [78]	Unsup.	75	**89**	-	-	**92**
Rubinstein [77]	Unsup.	63	48	**57**	**94**	70
VGG-19, K=1	Unsup.	65	41	49	43	49
ResNet-50, K=2	Unsup.	**77**	63	30	44	81
VGG-16	One-shot	68	51	31	46	32
VGG-16 BN	One-shot	60	60	44	43	50
VGG-19	One-shot	62	48	57	46	36
VGG-19 BN	One-shot	62	54	43	50	59
ResNet-50	One-shot	**77**	**73**	**74**	**66**	**88**

Table 4.2 – Object co-segmentation on five iCoseg image sets. We compare results across different levels of supervision. The colored cells of the unsupervised NMF approaches refer to the similarly colored factors shown in Figures 4.6, 4.7 (VGG-19) and 4.8, 4.9 (ResNet-50). Here, "Unsup." is unsupervised and "One-shot" refers to the use of a single ground truth instance, which we use to associate NMF factors with ground truth parts. In the one-shot case we consider factors from multiple factorizations, with $K = \{2,4,6,8\}$. Co-segmentation with NMF compares favorably against state-of-the-art methods, in spite of being based on low-resolution activations of a pre-trained network.

4.3.3 Layer depth

The NMF heatmaps considered so far have all been derived from activations of deep layers. In this section we study earlier layer in VGG-19 to evaluate the how much of the semantic information present in `conv5_4` is unique to that layer.

We can characterize the quality of *whole* factorization as the average IoU of its matching factors and parts (not including *background*). In Figure 4.10 we show the average IoU for different layers of VGG-19 on iCoseg with increasing K. The variance shown is due to repeated trials with different NMF initializations. There is a clear gap between convolutional blocks. Performance within a block, however, does not strictly follow the linear order of layers.

We also see that the optimal value for K is between 3 and 5. This is a result of not using factor merging in this experiment, which means factors must math the resolution of the part ground truth. As K increases, NMF heatmaps become more localized, highlighting regions that are beyond the granularity of the ground truth annotation, e.g., a pair of factors that separate *leg* into *ankle* and *thigh*.

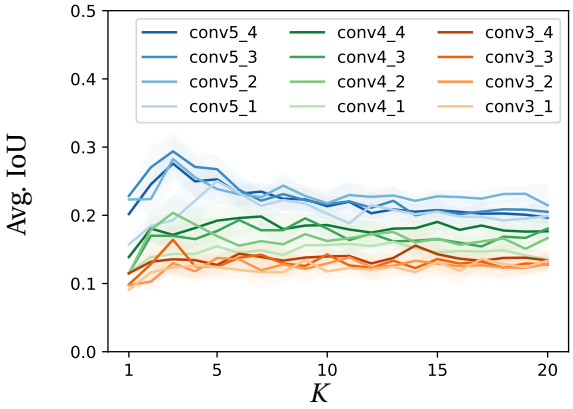


Figure 4.10 – Average IoU score for NMF with different layers of VGG-19 on iCoseg. As expected, earlier convolutional blocks match up significantly less to semantic parts.

4.4 Experiments on PASCAL VOC

PASCAL VOC has been commonly used to evaluate object co-localization methods. Images in this dataset often comprise several objects of multiple classes from various viewpoints, making it a challenging benchmark.

As in previous work [57; 19; 49], we use the *trainval* set for evaluation and filter out images that only contain objects which are marked as *difficult* or *truncated*. As we use PASCAL VOC

2007 specifically, the final set has 20 image sets (one per class), with 69 to 2008 images each.

For part co-segmentation, we use the PASCAL-Part dataset [18]. An extension of PASCAL VOC 2010 [32], this dataset has been further annotated with part-level segmentation masks and bounding boxes. The dataset decomposes 16 object classes into fine grained parts, such as *bird-beak* and *bird-tail* etc.1 After filtering out images containing objects marked as *difficult* and *truncated*, the final set consists of 16 image sets with 104 to 675 images each.

4.4.1 Object co-localization

The task of co-localization involves fitting a bounding box around the common object in a set of image. As before, with VGG-19 we set $K = 1$ to retrieve a heatmap which localizes that salient object across an image set.

After binarizing the single heatmap, as described in the previous section, we follow [83] and extract a single bounding box per heatmap. This is done by fitting a box around the largest connected component in the binary map.

We report the standard CorLoc score [25] of our localization. The CorLoc score is defined as the percentage of predicted bounding boxes for which there exists a matching ground truth bounding box. Two bounding boxes are deemed matching if their IoU score exceeds 0.5.

The results of our method are shown in Table 4.3. We compare against several state-of-the-art object co-localization methods. These methods operate by ranking set of object proposals, produced by a region-proposal CNN [62] or an object-saliency heuristic [19; 49]. They then choose the highest ranked region as a bounding box.

The authors of [57] present a method for unsupervised object co-localization that, like ours, also makes use of CNN activations. Their approach is to apply K-means clustering to globally max-pooled activations, with the intent of clustering all highly active CNN filters together. Their method therefore produces a *single* heatmap, which is appropriate for object co-localization, but cannot be extended to *part* co-localization.

Our method compares favorably to these previous approaches. For instance, we improve co-localization for the class *dog* by 16% higher CorLoc and achieve better co-localization on average, in spite of our approach being simpler and more general.

4.4.2 Part co-segmentation

In Table 4.4 we give IoU results for VGG-19 on five classes from PASCAL-Parts, which have been automatically matched to parts using all available ground truth data, as in section 4.3.1.

(a) Aeroplane

(b) Car

(c) Motorbike

Figure 4.11 – Example NMF heatmaps for three *vehicle* classes from PASCAL-Part with $K = 3$. We show four successful decompositions per-class and a failure case on the right-most column. NMF manages to retrieve interpretable decompositions in spite of great variation in the data.

(a) Bird

(b) Cow

(c) Cat

Figure 4.12 – Example NMF heatmaps for three *animal* classes from PASCAL-Part with $K = 3$. We show four successful decompositions per-class and a failure case on the right-most column. NMF manages to retrieve interpretable decompositions in spite of great variation in the data.

Method	aero	bicy	bird	boa	bot	bus	car	cat	cha	cow	dtab	dog	hors	mbik	pers	plnt	she	sofa	trai	tv	Mean
Joulin [49]	33	17	21	18	5	27	33	41	6	29	35	32	26	40	18	12	25	28	36	12	25.60
Cho [19]	50	43	30	19	4	62	**65**	43	**9**	49	12	44	64	57	15	9	31	34	62	**32**	36.60
Li [62]	**73**	45	43	28	7	53	58	45	6	48	14	47	**69**	67	24	13	**52**	26	65	17	40.00
Le (A) [57]	70	52	44	30	5	56	60	59	6	49	16	51	59	67	23	12	47	27	59	16	40.36
Le (V) [57]	72	**62**	48	28	**12**	**64**	59	72	6	37	12	45	67	72	19	11	37	29	**67**	23	41.97
VGG-19	63	49	54	20	10	62	51	**79**	4	51	32	**67**	67	73	19	15	43	35	66	24	44.20
ResNet-50	72	45	**60**	**42**	11	53	57	78	**9**	**60**	**36**	66	68	**74**	**26**	**18**	46	**55**	65	16	**47.98**

Table 4.3 – Co-localization results for PASCAL VOC 2007. Numbers indicate CorLoc scores. Using NMF with $K = 1$ applied to VGG-19 activations, we exceed the state-of-the-art approaches, though using a much simpler method.

In Figures 4.11 and 4.12 we visualize some of the corresponding $K = 3$ NMF heatmaps.

When comparing the heatmaps against their corresponding IoU-scores, several interesting observations arise. For instance, in the case of *motorbike*, the first and third factors for $K = 3$ in Table 4.4 both seems to correspond with wheel. The visualization in Figure 4.11b reveals that these factors in fact sub-segment the wheel into top and bottom, which is beyond the resolution of the ground truth data. The fact this distinction arises already with $K = 3$ indicates its importance in feature space, yet it is not an intuitively human distinction.

Returning to Table 4.4, when $K = 4$, a factor emerges that localizes instances of the class *person*, which occur in 60% of motorbike images. This again shows that while most co-localization methods only describe objects that are common across the image set, the NMF approach finds fine distinctions *within* the set of common objects.

Note that while the first factor of the class *aeroplane* (Figure 4.11a) consistently localizes airplane wheels, it does not to achieve high IoU due to the coarseness of the heatmap and fineness of the part.

In our final experiment, we use NMF heatmaps derived from both VGG-19 and ResNet-50 to segment the two classes, *cow* and *horse*. Since we have not come across examples of part co-segmentation in the literature, we compare against a method for supervised part segmentation, namely Wang and Yuille [96]. Their method relies on a compositional model with strong explicit priors w.r.t to part size, hierarchy and symmetry. We also show results for two baseline methods described in [96]: PartBB+ObjSeg where segmentation masks are produced by intersecting part-bounding-boxes [18] with whole-object segmentation masks [42]. The method PartMask+ObjSeg is similar, but here bounding-boxes are replaced with the best of 10 pre-learned part masks.

In spite of the highly domain-specific nature of their approach, we show in Table 4.5 that NMF nonetheless compares favorably to their results and even surpasses them in most cases using

K	aeroplane		bird		car		motorbike		cat	
1	*aeroplane*	42	*bird*	40	*car*	29	*wheel*	30	*eye/head/neck/nose*	31
2	*wheel*	2	*beak/eye/head/neck*	13	*wheel*	10	*wheel*	38	*torso*	24
	body/stern/tail/wing	49	*neck/torso/wing*	39	*door/roof/window*	22	*person*	9	*eye/head/neck/nose*	36
3	*wheel*	2	*leg*	2	*wheel*	10	*wheel*	30	*eye/head/neck/nose*	32
	body/stern/wing	47	*neck/torso/wing*	43	*door/headlight/licenseplat*	24	*headlight*	1	*torso*	30
	body/tail	35	*beak/eye/head/neck/tors*	30	*mirror/roof/window*	20	*wheel*	29	*ear/eye/head/neck/nos*	38
4	*wheel*	1	*foot/leg*	3	*wheel*	9	*wheel*	33	*eye/head/nose*	31
	body/wheel/wing	44	*neck/torso/wing*	44	*headlight/licenseplate*	31	*person*	10	*eye/neck/nose*	5
	stern/tail/wing	21	*beak/eye/head/neck/torso*	30	*front*	8	*wheel*	17	*ear/eye/head/nose*	35
	body/tail	32	*neck*	2	*mirror/roof/window*	22	*background*	13	*torso*	27

Table 4.4 – NMF on VGG-19 for PASCAL-Parts segmentation. Each NMF factor is automatically labeled with part labels as in section 4.3.1. Higher values of K allow NMF to localize finer regions across the image set, some of which go beyond the resolution of the ground truth part annotation. Figures 4.11 and 4.12 visualize the results for $K = 3$ (row color corresponds to heatmap color).

Method	*cow*			*horse*		
	head	*neck+torso*	*leg*	*head*	*neck+torso*	*leg*
PartBB+ObjSeg	26.77	53.79	11.18	37.32	60.35	27.47
PartMask+ObjSeg	33.19	56.69	11.31	41.84	63.31	21.38
Compositional model [96]	41.55	60.98	30.98	**47.21**	**66.74**	38.18
VGG-19	40.53	59.48	21.57	20.21	54.77	28.94
ResNet-50	**42.34**	**63.36**	**36.45**	34.31	64.49	**40.72**

Table 4.5 – Avg. IoU(%) for three fully supervised methods reported in [96] and for our NMF approach. Despite not using hand-crafted features, NMF compares favorably to these approaches, and is not specific to these two image classes. We manually matched NMF factors with their appropriate part labels by visually examining the heatmaps of only five images, out of approximately 140 images. This illustrates the usefulness of NMF co-segmentation for fast semi-automatic labeling. See visualization for *cow* heatmaps in Figure 4.12.

ResNet-50. This is in spite of not using any hand-crafted features or supervised training.

We note that for this experiment, our strategy for mapping NMF factors to their appropriate part labels was *manual*, in order to showcase the prospect of using NMF for *semi-automatic labeling*. We examined the heatmaps of five images, out of approximately 140 images, and manually assigned factors as corresponding to *head*, *tors+neck* or *leg*.

4.5 Conclusion

Following the result that well-generalizing CNNs are robust to NMF compression applied to their activations $\boldsymbol{A} \approx \boldsymbol{UV}$, in this chapter we studied the properties of the $\boldsymbol{U}$ matrix. By visualization $\boldsymbol{U}$ as a set of heatmaps, we could qualitatively see why this is the case: NMF factors correspond to semantic parts.

We evaluated the semantic quality of the resulting factors quantitatively with a series of co-segmentation and co-localization tasks, at the resolution of whole objects as well as finer parts. We found that in spite of their low resolution, the heatmaps derived from NMF factors match ground truth segmentation masks to high extent, with ResNet-50 showing better performance than some domain-specific over-engineered segmentation methods.

Based on these observation we hypothesize that CNNs learn clusters in feature space which correspond to a "natural" decomposition of the data into constituent parts. In a similar way to how Gabor-like edge detectors are optimal for sparse coding of images [71], and naturally emerges in both statistical models and the visual cortex, we propose that a decomposition of the kind detected by NMF is is optimal for the image-level classification task, and therefore naturally emerges in deep layers.

5 Semantic retrieval with matrix V

5.1 Introduction

In this chapter we continue our examination of the NMF factors, $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{V}$, focusing on the matrix $\boldsymbol{V}$. Recall that when $\boldsymbol{A} \in \mathbb{R}_{+}^{(N \cdot H \cdot W) \times C}$ is a CNN activation matrix, each row represents an *image patch embedding* in C-dimensional feature space. This space is a *latent* space, and as such it has no known a-priori interpretation. Its analysis and interpretation is the premise of our study of network interpretability.

The matrix $\boldsymbol{V}_{k,\cdot}$ represents a point or direction in the C-dimensional feature space. In the previous chapter we inferred the semantic meaning of $\boldsymbol{V}_{k,\cdot}$ via the *spatial distribution* of its corresponding $\boldsymbol{U}_{\cdot,k}$. Under a clustering interpretation of NMF, each $\boldsymbol{U}_{i,k}$ serves as a weight associating a datapoint $\boldsymbol{A}_{i,\cdot}$ with a cluster centroid $\boldsymbol{V}_{k,\cdot}$, as shown in Figure 5.1. Heatmap locations were considered matching when they showed strong weights with respect to the same centroid. For this reason, we think of the matrix $\boldsymbol{U}$ as encoding "*where*", and the matrix $\boldsymbol{V}$ as encoding "*what*", with the latter being represented in C-dimensional feature space.

We begin our study in Section 5.2 with a qualitative analysis, where we use gradient ascent to generate inputs that maximize activation in a direction $\boldsymbol{V}_{k,\cdot}$. While these visualizations are certainly interesting, we find they are not always relatable to the heatmaps derived from $\boldsymbol{U}$.

The matrix $\boldsymbol{V}$ is interesting because it represents the K most important non-negative directions in CNN feature space, with respect to reconstructing network activations. We can therefore think of $\boldsymbol{V}$ as a set of K *attributes*, indicating the presence of important semantic concepts within an image.

Some of the work presented in this chapter first appeared in [21].

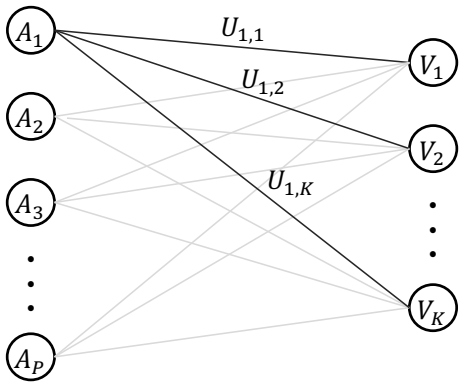


Figure 5.1 – Non-negative matrix factorization can be seen as forming a bipartite graph, where each data point $\boldsymbol{A}_i$ is associated with a component $\boldsymbol{V}_k$ by a weight $\boldsymbol{U}_{i,k}$.

It therefore gives us a means by which to compare different images without having to *jointly* factorize them. That is, instead of concatenating several images together, $\boldsymbol{A} = \left[\boldsymbol{A}^{(a)}\ \boldsymbol{A}^{(b)}\right]$, and factorizing against a single $\boldsymbol{V}$, we can factorize each image individually to obtain $\boldsymbol{V}^{(a)}$ and $\boldsymbol{V}^{(b)}$. If the rows of $\boldsymbol{V}^{(a)}$ and $\boldsymbol{V}^{(b)}$ are "similar", we can conclude the images are semantically related.

Based on this view, we quantify how well the rows of $\boldsymbol{V}$ describe the contents of individual images by performing *content-based image retrieval*. The goal of image retrieval in general is to correctly rank images from a large collection according to their *relevance* with respect to an input query. *Content-based* retrieval specifically means that the query itself is also an image.

Like in many other vision-related tasks, CNN-based methods currently hold the state of the art for content-based image retrieval. With deep CNN layers, moderately low-dimensional descriptors can be derived that encode global image semantics both succinctly and discriminatively. This allows for efficient content-based search, where a query image is matched against a large collection of images by a simple operation, e.g., cosine similarity.

Some of the best performing global descriptors are derived by aggregating several *local* descriptors. Selecting the appropriate local regions, however, is not straight forward. Methods have been proposed that simply consider the whole image as single region [4; 5]. Others impose a simple grid over the feature maps [91] and some even randomly sample regions [85]. In our case, viewing $\boldsymbol{V}$ as a set of local descriptor sets their corresponding regions to nothing else but the heatmaps encoded in $\boldsymbol{U}$.

Most CNN-based methods for image retrieval focus specifically on *instance-based image retrieval*, where the goal is to retrieve images that contain the same object instance as the query image. For example, given an image of a building, we aim to highly rank other images of that specific building, and not images of similar buildings.

A related task is that of *semantic image retrieval* [38] where given a query image, e.g., of a *dog*, we aim to rank highly all images in our collection that portray any *dog*. This task introduces much greater variability into the set of relevant images, and limits an algorithm's reliance on surface features such as texture and color.

In Sections 5.3 and 5.4 we tackle these two tasks. Interestingly, while we show in Section 5.3 that a descriptor based purely on $\boldsymbol{V}$ excels at semantic image retrieval, it is insufficient for instance-based image retrieval. Instead, the information required to accomplish the latter task is distributed across both $\boldsymbol{U}$ and $\boldsymbol{V}$. As we show in Section 5.3, when both are combined, our algorithm yields state-of-the-art results also for instance-based image retrieval.

5.2 Gradient ascent visualization

As described in Section 2.2.4, several methods have been proposed that directly visualize directions in CNN feature space. This is accomplished by generating an input image that maximizes activation in that direction. Since a row $\boldsymbol{V}_{k,\cdot}$ inhabits this space, it is interesting to see if these visualization agree with our interpretation via the heatmaps derived from $\boldsymbol{U}_{\cdot,k}$.

We used to the method of Olah et al. [70], where we optimized:

$$\mathsf{I}_k = \arg\max_{\mathsf{I}} \sum_{i,j} \cos\left(\mathsf{A}_{\cdot,i,j}, \boldsymbol{V}_{k,\cdot}\right) \tag{5.1}$$

where $\mathsf{A} \in \mathbb{R}_+^{C\times H\times W}$ is the deep layer activation in response to input I_k. In their method, the image I_k is parameterized by its Fourier coefficients. This parameterization guides the optimization with gradient ascent towards qualitatively more pleasing visualizations.

In Figures 5.2 and 5.3 we show gradient ascent visualization obtained for the $\boldsymbol{V}$ derived from activation of VGG-19 on the iCoseg dataset. Specifically, we return to the examples of Figures 4.6 and 4.7, where we applied NMF with increasing rank K to the activations of two image sets, *Elephants* and *Gymnastics1*. Whereas in Figures 4.6 and 4.7 we visualized the NMF $\boldsymbol{U}$ matrix as heatmaps, here we visualize the $\boldsymbol{V}$ matrix as described above. The colored frames in Figures 5.2 and 5.3 correspond to the similarly colored heatmap derived from $\boldsymbol{U}$.

We created visualization with ResNet-50 as well, shown in Figures 5.4 and 5.5. These differ substantially from those of VGG-19, and are dominated by color and texture. With the exception of some background components resembling *trees*, these visualization do not suggest any coherent structures or objects.

As can be seen, the quality of the visualization differs significantly between examples. While in Figure 5.2 one can readily discern *elephant*-like parts reflected in the visualization, Figure 5.5

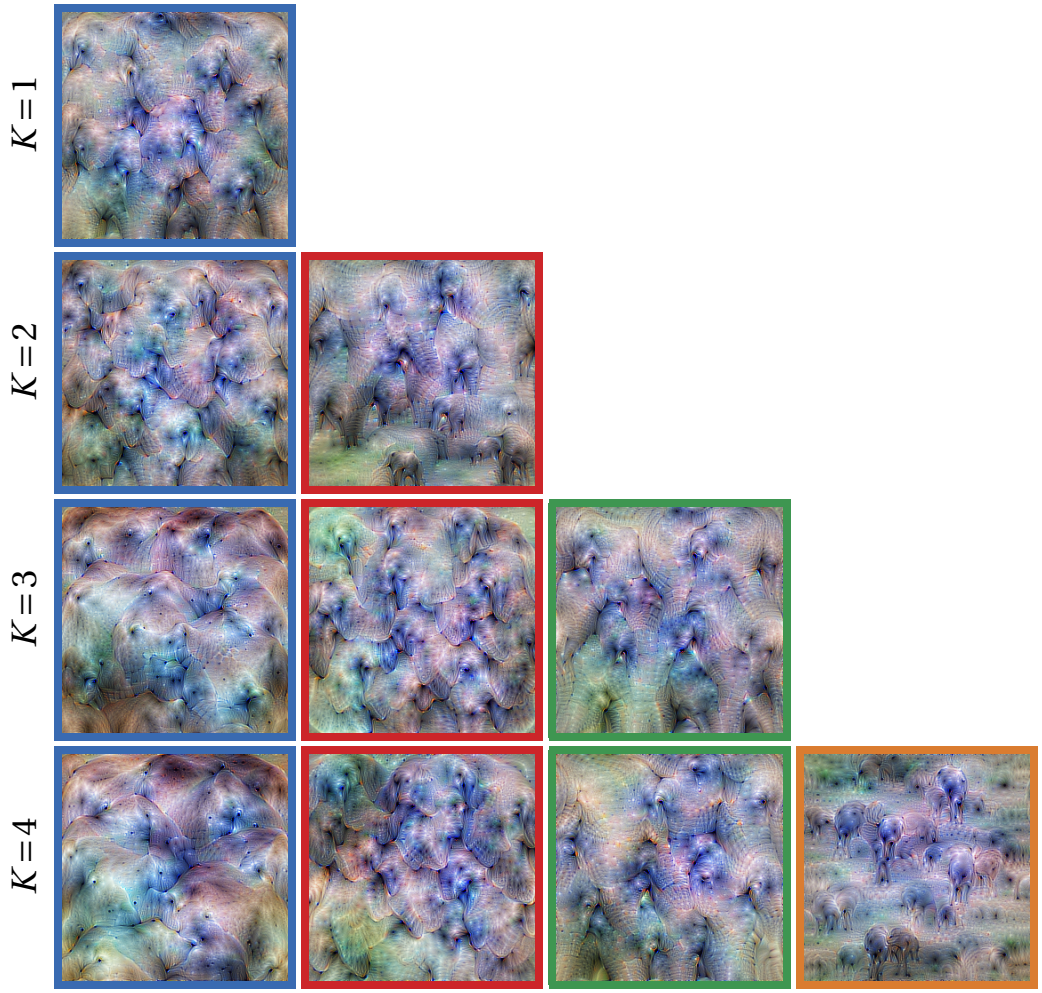


Figure 5.2 – Gradient ascent visualization of the NMF basis vectors derived from VGG-16 on the *Elephants* subset from iCoseg. These visualizations of the rows of $\boldsymbol{V}$ correspond to the heatmaps derived from the columns of $\boldsymbol{U}$ shown in Figure 4.6 using the same color encoding, i.e., the blue framed visualizations above correspond to the blue heatmaps in each corresponding row.

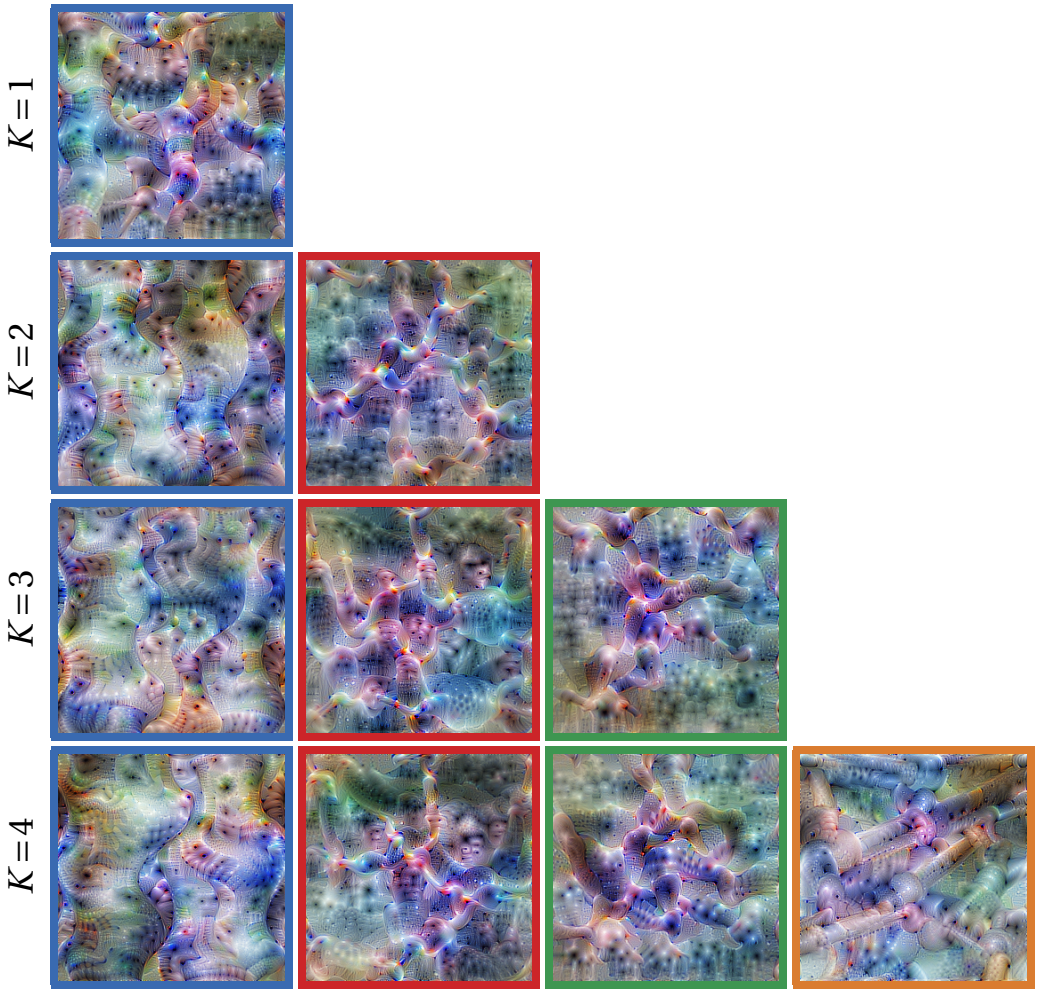


Figure 5.3 – Gradient ascent visualization of the NMF basis vectors derived from VGG-16 on the *Gymnastics1* subset from iCoseg. These visualizations of the rows of $\boldsymbol{V}$ correspond to the heatmaps derived from the columns of $\boldsymbol{U}$ shown in Figure 4.7 using the same color encoding, i.e., the blue framed visualizations above correspond to the blue heatmaps in each corresponding row.

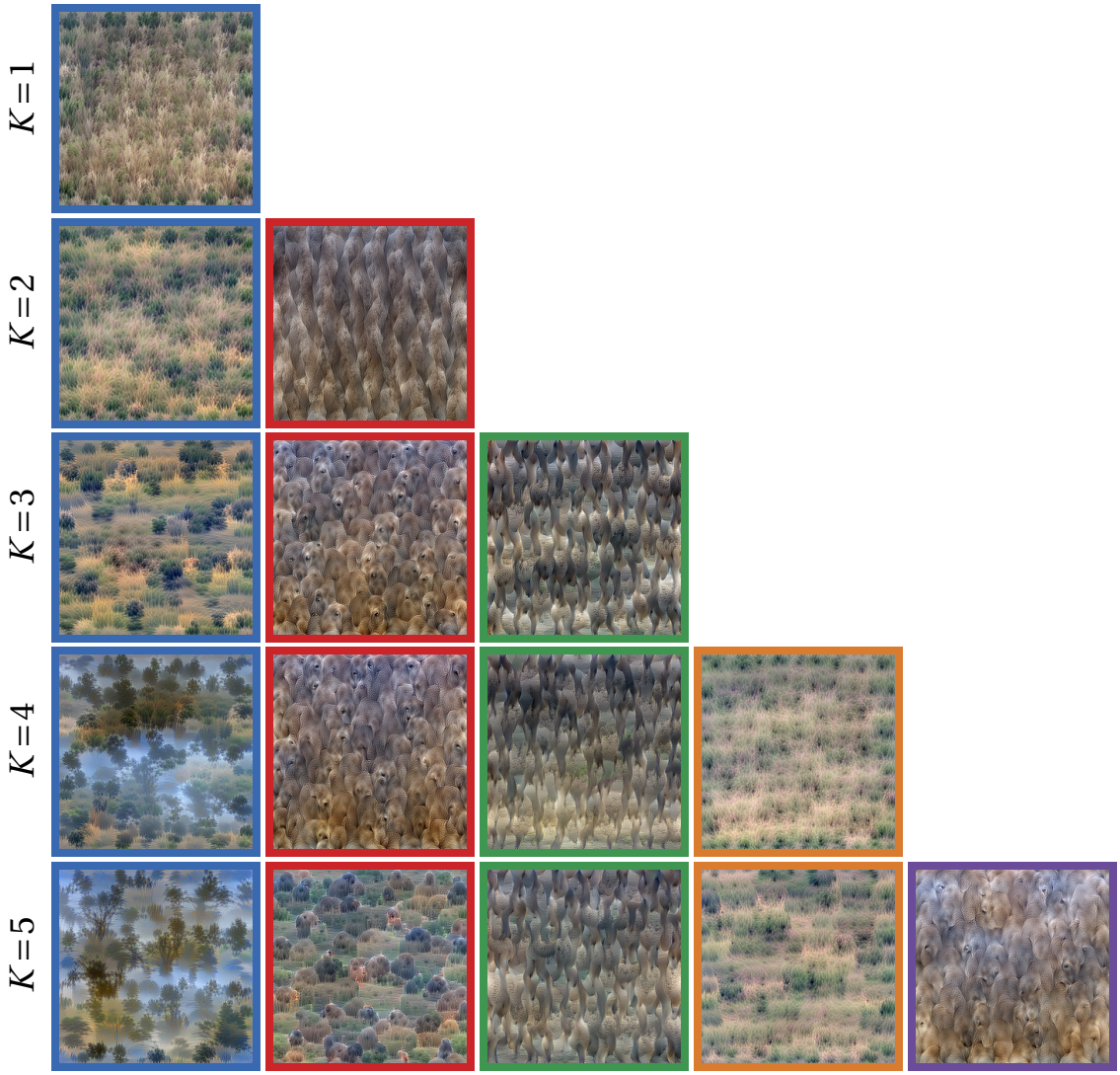


Figure 5.4 – Gradient ascent visualization of NMF basis derived from ResNet-50 on *Elephants* subset from iCoseg. These visualizations of the rows of $\boldsymbol{V}$ correspond to the heatmap visualizations of the columns of $\boldsymbol{U}$ shown in Figure 4.8 using the same color encoding, i.e., the blue framed visualizations above correspond to the blue heatmaps in each corresponding row.

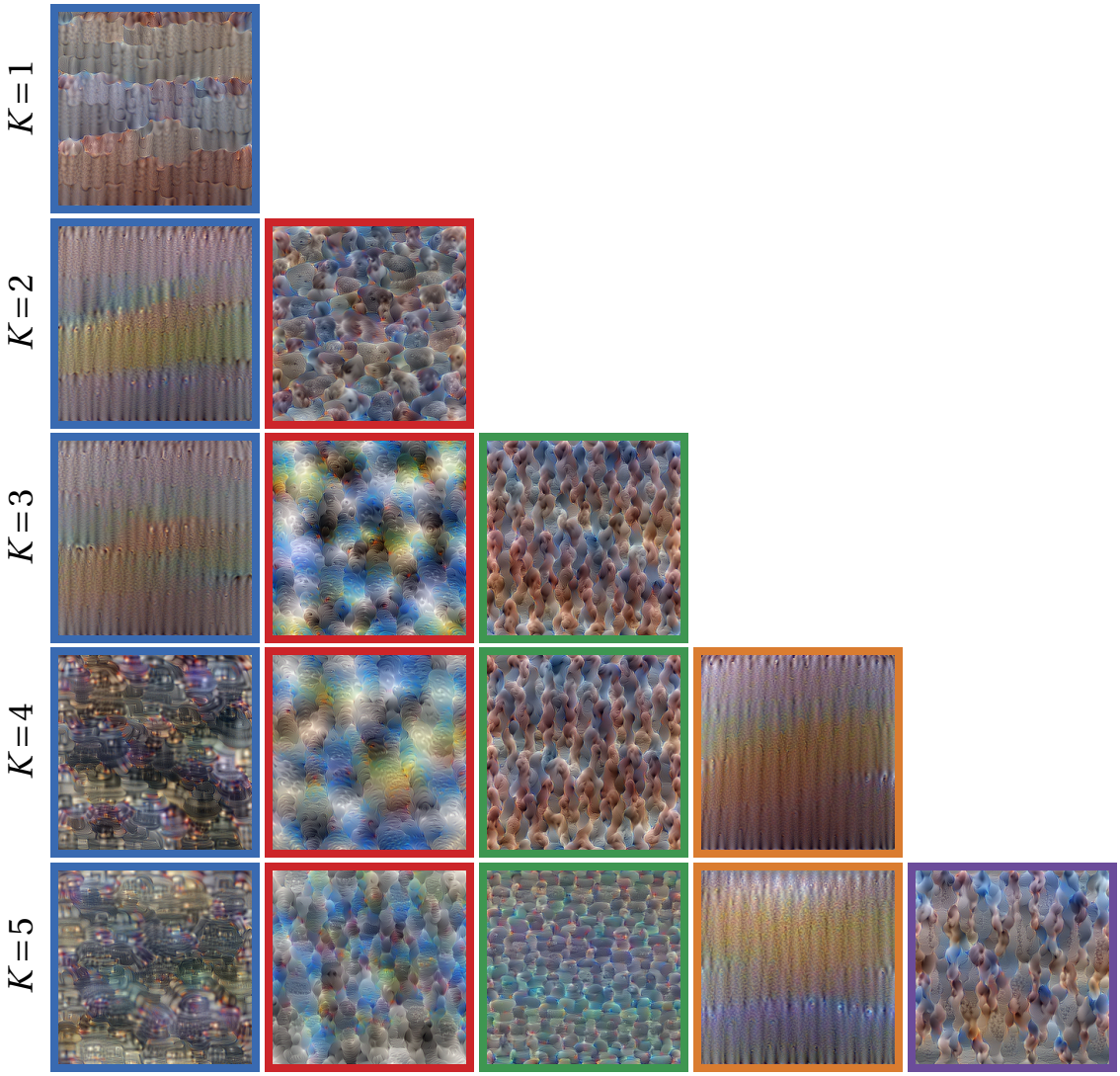


Figure 5.5 – Gradient ascent visualization of NMF basis derived from ResNet-50 on *Gymnastics1* subset from iCoseg. These visualizations of the rows of $\boldsymbol{V}$ correspond to the heatmap visualizations of the columns of $\boldsymbol{U}$ shown in Figure 4.9 using the same color encoding, i.e., the blue framed visualizations above correspond to the blue heatmaps in each corresponding row.

is quite more abstract and not easy to interpret.

5.3 Experiments on Oxford and Paris buildings

We evaluate NMF for image retrieval using two standard datasets, Oxford Buildings [72] and Paris Buildings [73]. These datasets consist of 5,063 and 6,392 images, respectively, and each includes 55 query images with a bounding box annotation and a list of corresponding relevant matches. We process images by scaling their smaller side to 512 pixels, and perform pixel-wise normalization to achieve zero mean and unit variance. In Figures 5.6 and 5.7 we show two example subsets from Paris Buildings overlaid with NMF heatmaps.

Given the activation tensor for a single image $\boldsymbol{A} \in \mathbb{R}_+(N \cdot H \cdot W) \times C$, our goal is to obtain for each image a C-dimensional vector. We call this vector the global NMF descriptor of the image, and denote it as $\boldsymbol{v}$.

After we obtain $\boldsymbol{v}$, we score the match between query and target descriptors using cosine similarity. The ranking of the returned images is evaluated using mean average precision (mAP), defined as follows. First we define the precision at ranking position n:

$$\text{P@n} = \frac{r}{n} \tag{5.2}$$

where r is the number of relevant images in the top n results.

Now let $\mathbb{1}_n$ be an indicator function which is 1 one the nth result is relevant. Furthermore let our image collection consist of N images, of which R are relevant and for the current query. We define:

$$\text{AP} = \frac{\sum_n^M \mathbb{1}_n \text{P@n}}{R} \tag{5.3}$$

the *mean* average precision is computed by averaging the average precision over all queries in the query set.

5.3.1 Instance-based retrieval

We now describe how we derive $\boldsymbol{v}$ from $\mathbf{A}$. An overview of all steps is shown in Figure 5.8.

Before applying NMF, we adopt a method proposed by [50], known as CroW. This method weighs each channel $1 \leq c \leq C$ with a weight $\mathbf{W}_c$. This weight decreases inversely with the

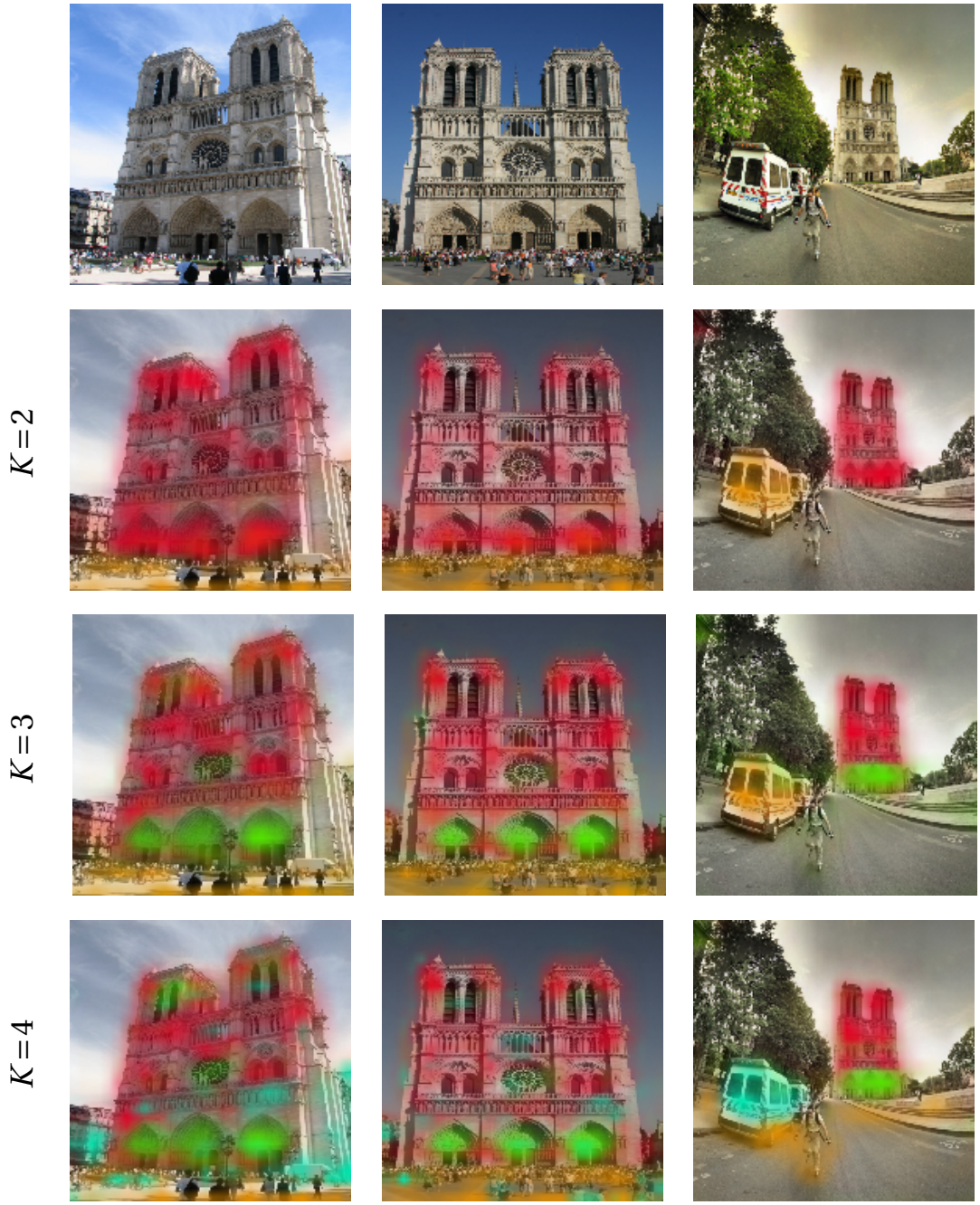


Figure 5.6 – Three queries of the *Notre-Dame* subset of Paris Buildings, shown here with NMF heatmaps (matrix $\boldsymbol{U}$) derived using VGG-19.

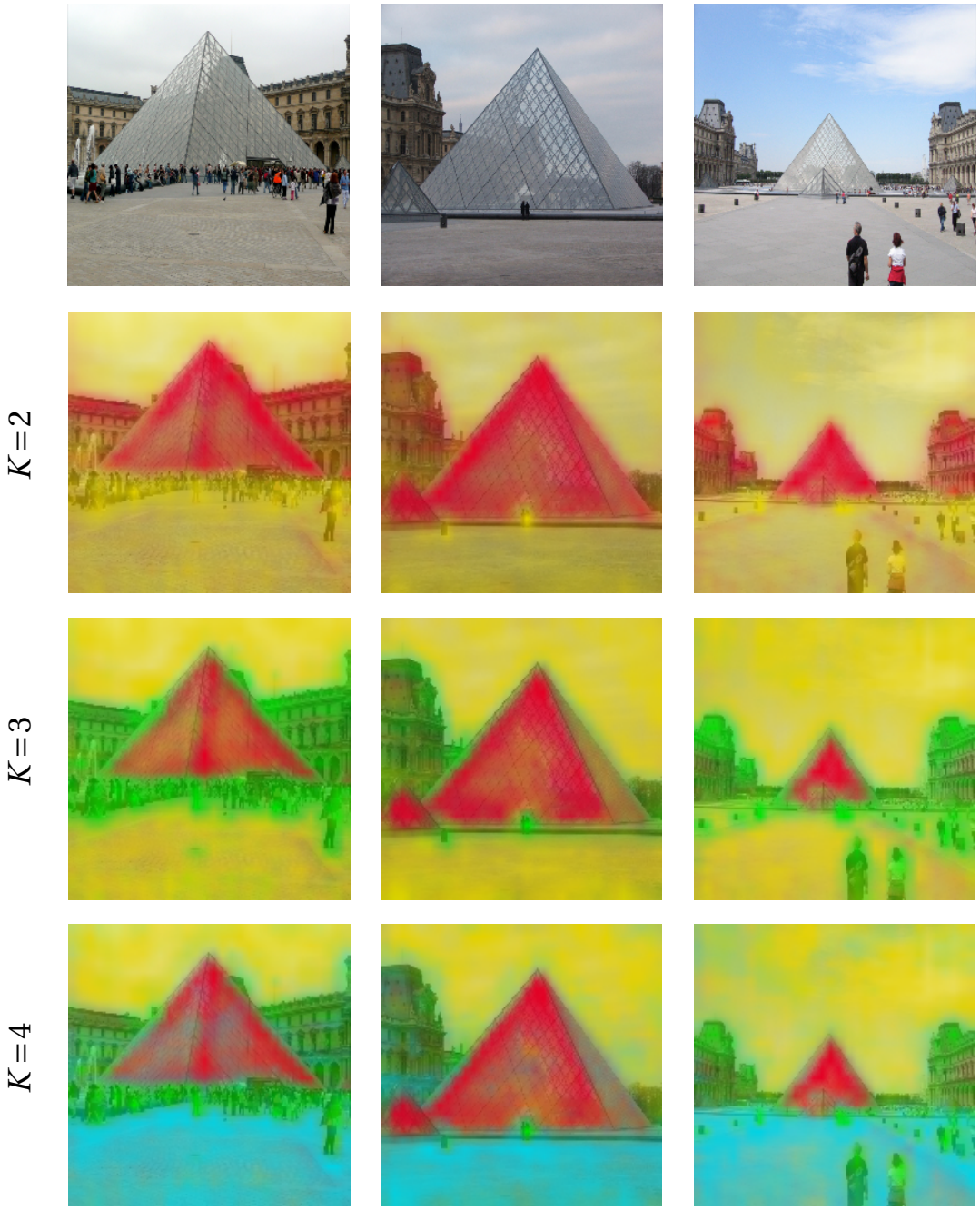


Figure 5.7 – Three queries of the *Louvre* subset of Paris Buildings, shown here with NMF heatmaps (matrix $\boldsymbol{U}$) derived using ResNet-50.

ratio of positive channel activations, Q_c:

$$Q_c = \frac{\sum_{w,h} \mathbb{1}\left[\mathbf{A}_{c,w,h} > 0\right]}{H \cdot W} \qquad \mathbf{W}_c = \log\left(\frac{\sum_{c'}^{C} Q_{c'}}{Q_c}\right) \tag{5.4}$$

where again $\mathbb{1}[\cdot]$ is an indicator function that equals 1 when its condition evaluates to *true*.

As a result, in $\mathbf{A}' = \mathbf{A} \cdot \mathbf{W}$, the contribution of channels that are relatively less active is increased. That reasoning behind this procedure is analogous to that of *tf-idf* (term frequency–inverse document frequency) often used in the context of document retrieval. This weighing amplifies rare features, since these are more likely to be discriminative than features that are very common.

Next, we obtain the NMF matrices $\boldsymbol{U}$ and $\boldsymbol{V}$. At this point of the pipeline we have the option to incorporate spatial information from $\boldsymbol{U}$, in addition to the semantic information in $\boldsymbol{V}$. This will be discussed later on, but for now we proceed considering only the matrix $\boldsymbol{V}$.

Viewing the rows of $\boldsymbol{V}$ as local descriptors, we follow a pipeline that has become standard practice in image retrieval pipelines. Namely, we normalize each local descriptor, perform PCA-whitening, followed by row-wise summation and a final $l2$-normalization. The resulting *vector* is $\boldsymbol{v} \in \mathbb{R}^C$, the global NMF image descriptor.

At search time, a query image is given along with a bounding box indicating the region of interest. In this case, we follow a similar procedure but include an additional step to filter out rows of $\boldsymbol{V}$ that represent irrelevant concepts. In particular, since each row of $\boldsymbol{V}$ has a corresponding heatmap, we use the provided bounding box to select a subset $\boldsymbol{V}_f \in \mathbb{R}_+^{K' \times c}$ of K' factors, whose heatmaps allocate more than 75% of their activation within the box. This is reminiscent of our "one-shot" co-segmentation procedure in Section 4.3.1. Note that this use of the query bounding box is different from how other approaches to image retrieval use it, where the box is typically used to crop the query image. The matrix $\boldsymbol{V}_f$ replaces $\boldsymbol{V}$ in the remainder of the NMF descriptor extraction pipeline as described thus far.

The results of this pipeline are given in Table 5.1 as NMF $\boldsymbol{V}$, since we only used $\boldsymbol{V}$ to derive the NMF descriptor. We compare our results against several baseline and state-of-the-art methods.

Namely, R-MAC [91] (Regional Maximum Activation of Convolutions) proceeds by sliding several max-pooling kernels over the $\mathbf{A}$, each with a different scale and aspect ratio. At every spatial position max-pooling generates a C-dimensional vector, which is stored. The result is a matrix with a number of rows which depends on the size of $\mathbf{A}$ and C columns. The rows are

then identically as $\boldsymbol{V}$ in the pipeline we have described for NMF.

More recently, [47] proposed image descriptors derived using class-activation maps (CAM) [103] (see Section 2.2.4). Similarly to NMF heatmaps, CAM maps define a soft spatial weighting over the feature map, dividing it into several semantically meaningful regions. There are, however, two downsides to CAM in this context. First, CAM requires a specialized CNN architecture, and is not compatible with just any off-the-shelf pre-trained CNN. Second, CAM depends on last layer outputs, and thereby on the set of output labels used for CNN training. NMF is on the other hand is compatible with any ReLU CNN, and is not bound to any label set.

Returning to Table 5.1, while performance with VGG-16 is improved compared to competing methods, and overall performance with NMF is superior to retrieval with CAM, on ResNet-50 our method does not do as well as R-MAC. The same is true also when we perform *query expansion*, i.e., we aggregate the descriptors of the top-5 matching results together with the original query descriptor, re-normalize, and repeat the search.

To understand where our method fails, consider Figure 5.9 where we show a query from Oxford Buildings along with its top five matches, two of which are in fact irrelevant. The query image is marked by an orange frame, and green and red frames indicate relevant and irrelevant matches respectively. To better understand how NMF sees these images, we *separately* factorized each image with $K = 4$, as also shown in 5.9, where we aligned similar factors in the same row for easier inspection. When examining the false positives, it is difficult to find a clear fault in the heatmap correspondence. Components such as *window* and other architectural components are indeed shared between the query image and the false positives. The difference lies not in their presence or absence, but in their spatial arrangement.

To account for this, we return to the matrix $\boldsymbol{U}$ and incorporate it into the NMF descriptor in the following way, First, we reshape $\boldsymbol{U}$ into a tensor $\mathsf{U} \in \mathbb{R}^{K \times H \times W}$, and compute:

$$\boldsymbol{U}_G = \text{matrix}(\mathsf{U} * \boldsymbol{G}) \tag{5.5}$$

$$\boldsymbol{U}'_{\cdot,j} = \frac{[\boldsymbol{U}_G]_{\cdot,j}}{\|[\boldsymbol{U}_G]_{\cdot,j}\|_2} \tag{5.6}$$

$$\boldsymbol{V}' = \boldsymbol{U}'^{\top}\boldsymbol{U}'\boldsymbol{V} \tag{5.7}$$

where $\boldsymbol{G}$ is a 2D Gaussian filter, convolved against each of K heatmaps in U. The matrix $\boldsymbol{U}'^{\top}\boldsymbol{U}'$ is a $K \times K$ matrix capturing *spatial interactions* between concepts. Multiplication against this matrix informs the resulting $\boldsymbol{V}'$ of which concepts are adjacent to which. We empirically set the Gaussian filter $\boldsymbol{G}$ to have a kernel size of 13×13 pixels and a variance of 2.

With the rest of the pipeline unaltered, we repeat the experiment and report the results in 5.1 as NMF $\boldsymbol{V} + \boldsymbol{U}$. As can be seen, with this additional information, NMF descriptors significantly

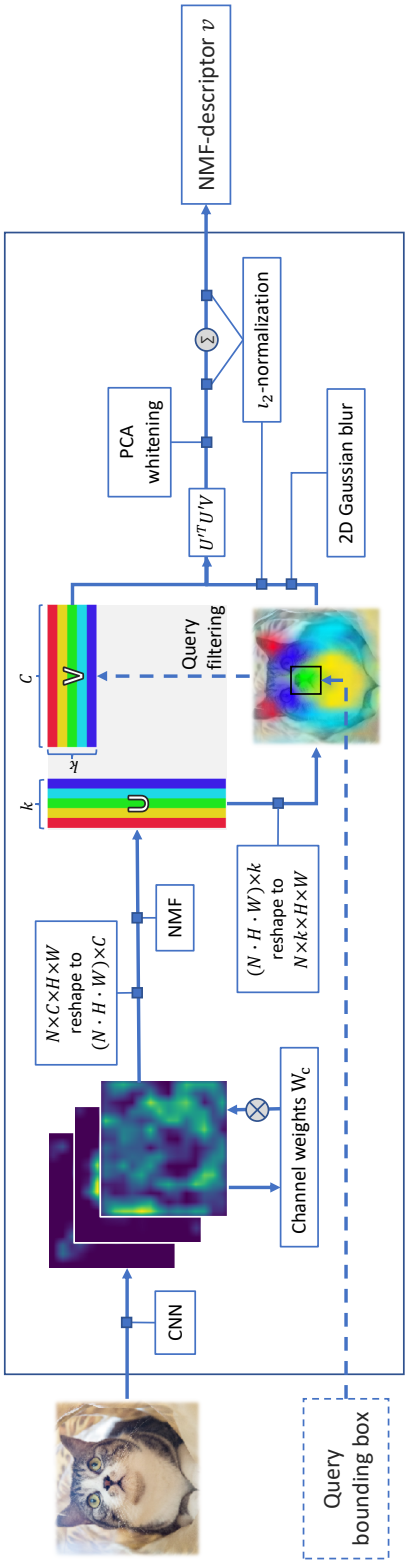


Figure 5.8 – NMF descriptors are obtained by aggregating NMF basis vectors of deep CNN feature activations. Shown here in solid lines is the pipeline used to extract a descriptor for every image in the large collection being indexed. An activation tensor representing N images is reshaped into a matrix whose column represent the C channels at that layer (e.g. 512 for VGG-16 final convolutional layer). Using NMF, the matrix is decomposed into a predefined small number of factors, K. When reshaped back into feature maps, the K columns of $\boldsymbol{U}$ form heatmaps, which highlight semantic concepts in the scene, while rows in $\boldsymbol{V}$ are descriptors of those concepts in CNN feature space. The matrices $\boldsymbol{U}$ of $\boldsymbol{V}$ are combined to form the NMF query descriptor, as discussed in section 5.3.1. At search time, we additionally employ the pipeline shown using a dashed line. Before aggregation, a bounding box is used to filter out rows of $\boldsymbol{V}$ whose corresponding heatmaps do not fall within the bounding box. Filtering thus retains only factors that are likely to be relevant to the query.

Figure 5.9 – An example of incorrect ranking with using NMF $\boldsymbol{V}$ for instance-based retrieval. The query image is on the left surrounded by a yellow frame. To its right, a green frame signifies a relevant image and a ref frame signifies an irrelevant image. While all top scoring matches with NMF $\boldsymbol{V}$ do in fact share common elements with the query image, they do not depict the same instance.

outperform all other methods.

5.3.2 Localization

Given the set of top matching images with respect to some query, it is straight forward to localize the relevant image regions by applying NMF jointly, as in Chapter 4. This generates corresponding heatmaps extending both the query and its matches. As before, we keep only those heatmaps that are sufficiently contained within the query bounding box.

Two examples are shown in Figure 5.10, where we re-scaled images to be 224 pixels on the smaller side, followed by NMF and a bounding box prediction. The predicted bounding box (shown in red) is obtained in three steps. First, the filtering step discards NMF maps

	Oxford5k		Paris6k	
	VGG	ResNet	VGG	ResNet
MAC [4]	55.7	57.2	68	69.9
R-MAC [91]	67.8	71	77.4	81.4
CroW [50]	65.4	63.3	74.3	71.7
CAM* [47]	71.2	69.9	80.5	80.4
NMF $\boldsymbol{V}$	71.9	67.7	81.3	77.5
NMF $\boldsymbol{V}+\boldsymbol{U}$	**73.4**	**74.8**	**83**	**83.2**

(a)

	Oxford5k		Paris6k	
	VGG	ResNet	VGG	ResNet
MAC [4]	60.2	66.5	77.6	81.2
R-MAC [91]	72.1	77.5	82.4	85.5
CroW [50]	68.5	68	77.1	76.4
CAM* [47]	73	-	83.6	80.4
NMF $\boldsymbol{V}$	73.8	69.2	82.3	81.4
NMF $\boldsymbol{V}+\boldsymbol{U}$	**74.2**	**78.8**	**83.8**	**86.2**

(b)

Table 5.1 – Instance-based retrieval mAP results for various state-of-the-art methods and our NMF approach. In (a) we show single-pass retrieval results and in (b) after top-5 query expansion *To enable CAM, the authors used a fully-convolutional variant of VGG-16 was used instead of the standard VGG-16 architecture.

which allocate less than 75% of their activation within the query bounding box. Next, the remaining heatmaps are averaged to form a single map, which is binarized by setting the top 30% activations to one, and the rest to zero. Finally, a bounding box is placed around the largest connected component.

In [91] localization is performed using R-MAC, i.e. using region max-pooling. First, the query image is cropped according to its bounding box, and its MAC descriptor is obtained. Then the query descriptor is scored with a similarly extracted local descriptor computed for *every* window in each of the target images. A more efficient approximate method, AML, is also proposed. We compare their reported localization results on Oxford buildings and Paris Buildings to ours in Table 5.2.

Our evaluation follows the same protocol of [91], cross matching the five query images that are given per building. A single image is used as a query (orange box), and the resulting predictions (red boxes) are evaluated on the other four (green boxes). This is repeated using each of the five images as a query. Bounding box overlap is measured with intersection over union (IoU).

NMF achieves better localization results, and again we find ResNet-50 outperforming VGG-16.

(a) VGG-16 localization (b) ResNet-50 localization

Figure 5.10 – NMF heatmaps localize objects within retrieved images, as shown in the two examples above. In each example, we apply NMF to the image set and obtain K factors. The ground truth bounding boxes for the region of interest are shown in orange and green. In each row, using only the orange bounding box, we filter out factors whose heatmap is not localized within the box. The remaining K' heatmaps are averaged to form a single heatmap per image, shown above. The bounding boxes shown in red are predicted by binarizing the averaged heatmap, and surrounding the largest connected component. When compared to ground truth boxes *not* used for filtering, i.e., only those shown in green, our predictions overlap substantially. See Table 5.2 for quantitative results. Best viewed on a color display.

Method	Oxford5k	Paris6k
AML [91]	51.3	51.4
Exhaustive R-MAC [91]	52.6	52.9
NMF VGG-16	49.3	67.6
NMF ResNet-50	**53.2**	**68.7**

Table 5.2 – Bounding box IoU for NMF on query images. As shown in Figure 5.10, we can apply NMF to a query and its matches to retrieve semantically corresponding regions across the image set. By using the bounding box of the query image, we select only the regions relevant to the query. We surround the relevant factors in the remainder of the image set with a predicted bounding box.

5.4 Semantic image retrieval on PASCAL VOC

In the previous section we found that an image descriptor derived solely based on $\boldsymbol{V}$ was sub-optimal for instance-based image retrieval. We explained this with $\boldsymbol{V}$ indicating the presence or absence of certain semantic concepts, it lacked information about the spatial relationship between them, which is needed to discriminate between instances of the same category. If true, however, then this makes a descriptor extracted based solely on $\boldsymbol{V}$ highly suitable the related task of *semantic* image retrieval, where the objective is to retrieve all instances of the class.

In this section we verify this hypothesis by evaluating NMF, and other methods introduced in the previous section, on retrieval with PASCAL VOC 2010 [32]. Like other version of PASCAL VOC, the 2010 version consists of 20 classes, mostly of animals, vehicles and furniture. We filtered out images whose main object was labeled *difficult* or *truncated*, which left us with a total of 5,455 images, ranging from 86 (*dining table*) to 980 (*person*).

We then manually selected five query images from each category for a total of 100 queries. The set of relevant images consisted of all images containing the main object in the query. To stay consistent with the retrieval methodology above, we needed to supply a bounding box surrounding the main query object. Since images in PASCAL VOC can contain multiple instances of the same category, we selected a single bounding box by first merging any overlapping bounding boxes and then choosing the single largest box.

We evaluated that quality of the retrieved ranking with *mean average precision at 30* (mAP@30):

$$\text{AP@30} = \frac{\sum_{n}^{30} \mathbb{1}_n \text{P@n}}{30} \tag{5.8}$$

This choice avoids issues due to class imbalance present in this dataset, and also reflects the fact that users, in general, do not care about the long tail of search results, but rather focus on

Network	Method	aero	bicy	bird	boa	bot	bus	car	cat	cha	cow	dtab	dog	hors	mbik	pers	plnt	she	sofa	trai	tv	Mean
ResNet-50	MAC [4]	13.5	22.4	10.3	6.7	5.6	9.6	14.8	7.7	5.6	4.6	3.9	18.1	9.2	9.5	24.8	9.0	10.1	3.5	6.9	13.9	10.5
	CroW [50]	32.5	37.3	17.6	12.9	11.2	23.6	36.8	16.3	11.3	7.6	10.5	27.6	19.2	25.0	43.3	16.9	8.6	10.0	7.7	21.6	19.9
	R-MAC [91]	38.1	48.4	30.8	14.8	15.2	27.0	21.6	26.9	13.2	6.5	7.2	33.2	28.1	24.7	38.4	23.3	21.4	10.3	11.6	22.9	23.2
	NMF $\boldsymbol{V}+\boldsymbol{U}$	71.3	56.4	46.4	43.5	37.8	57.6	72.7	40.0	25.3	13.3	20.0	43.9	39.9	33.3	60.0	30.5	31.5	32.7	34.2	41.6	41.6
	NMF $\boldsymbol{V}$	91.9	83.4	54.2	53.1	53.2	72.8	85.4	58.0	31.6	11.9	26.0	53.7	47.2	60.3	89.1	34.7	38.2	42.3	35.8	43.5	53.3
VGG-16	MAC [4]	43.4	42.9	42.5	24.0	30.8	49.5	46.9	38.9	14.4	11.1	5.9	57.6	22.7	22.9	48.1	23.8	31.0	11.3	24.4	29.3	31.1
	CroW [50]	76.5	69.0	58.8	38.6	35.6	64.9	64.0	52.6	29.1	22.3	17.1	64.8	54.3	56.4	71.1	33.6	49.1	41.2	37.8	54.9	49.6
	R-MAC [91]	83.5	82.5	70.3	49.3	49.2	69.4	71.9	57.0	28.1	19.0	14.6	59.9	53.6	54.7	74.7	55.2	62.9	39.5	53.7	64.6	55.7
	NMF $\boldsymbol{V}+\boldsymbol{U}$	96.4	67.6	64.3	62.9	64.6	80.7	87.9	84.1	29.3	25.2	28.6	74.4	71.8	67.4	76.7	52.8	70.7	43.8	42.8	81.5	63.7
	NMF $\boldsymbol{V}$	**100**	**97.5**	**83.5**	**82.6**	**72.5**	**92.5**	**95.3**	**90.1**	**44.6**	**31.0**	**29.4**	**86.0**	**82.6**	**94.2**	**81.2**	**71.9**	**86.8**	**51.1**	**56.0**	**95.8**	**76.2**

Table 5.3 – Semantic image-retrieval on PASCAL VOC 2010 using ResNet-50 and VGG-16. Unlike instance-based image retrieval, the "clean" semantics of the rows of $\boldsymbol{V}$ yield better results than descriptor which partially encode mixing between different semantic concepts. Furthermore, unlike localization-based tasks, here VGG-16 significantly outperforms ResNet-50. This results would suggest that ResNet-50 activation still hold a fair degree of surface features, which while conducive for precise localization, is less invariant to nuisance variables.

the top few. The descriptor extraction pipeline is identical to that of the previous section, for all methods.

We report results in Table 5.3. As expected, the descriptors based on NMF $\boldsymbol{V}$ outperform both R-MAC and NMF $\boldsymbol{V}+\boldsymbol{U}$. Also notable is the excellent performance of VGG-16 compared to ResNet-50. These results, along with other empirical results presented throughout this thesis, suggest that ResNet-50 features hold a larger degree of low-level information, e.g., about texture and color. While this is conducive for precise localization, these features are less invariant to nuisance variables, and evidently perform worse on the task of semantic image retrieval.

5.5 Conclusion

Just as the columns of $\boldsymbol{U}$ represent semantic concepts by virtue of their spatial distribution, the rows of $\boldsymbol{V}$ represent those same concepts in CNN feature space.

In this chapter we leveraged this property to derive fixed-size global image descriptors that at once characterize the potentially many concepts present within an image, while remaining discriminative with respect to them.

We again evoked the over-simplified, yet useful, view of the NMF factors representing "*where*" and "*what*", which led to two flavors of image descriptors, which are useful in different in contexts.

Relevant to our discussion of future work in the next section, we briefly mention here that image retrieval can be further improved with end-to-end CNN training [39; 75]. We limited ourselves to the setting of using a pre-trained network, since our goal was to study the networks and their NMF decomposition. Nonetheless, the NMF descriptors we derived outperformed all other methods in this category.

6 Conclusion

6.1 Thesis summary

We started this thesis with the goal of understanding what distinguishes neural networks that learn from those that simply memorize their training set. In Chapter 2 we formulated memorization as the information $\mathcal{I}(\mathbf{i}, \mathbf{Z})$ that an intermediate NN representation $\mathbf{Z}$ holds about a specific input, index by $\mathbf{i}$. We showed that this quantity is upper bounded by the *non-negative rank* of a matrix, $\boldsymbol{A}$, in which the ith row contains the intermediate NN representation, i.e., $\boldsymbol{A}$ is the activation matrix.

Since the non-negative rank is NP-hard to compute, we derived a related quantity, computable in reasonable time, that allowed us to compare the amount of memorization across different networks, as well as predict which one will generalize better to new data. Specifically, using *approximate* NMF, $\boldsymbol{A} \approx \boldsymbol{U}\boldsymbol{V}$, with different rank constraints K, we showed that the area under the K vs. classification accuracy curve serves as a good proxy to the non-negative rank.

Using NMF as well as other matrix factorization methods, we found that NNs that memorize less are more robust to compression applied to their activations, indicating the underlying data manifold is intrinsically lower-dimensional, compared to NNs that memorize more. Conversely, we showed that networks that memorize less are more vulnerable to their activations being *ablated* in the directions found by by NMF. During NN training, we demonstrated that NMF can guess when to *early stop* quite precisely, sparing the use of a validation set.

In Chapter 4 we examined the matrix $\boldsymbol{U}$ generated by NMF when applied to deep CNN activations, and saw that it provided a rare and interpretable view into the emergent semantics encoded in deep CNN layers. The fact that we sought a means to evaluate memorization and generalization and found that it provides a useful tool for network interpretability outlines that generalization and interpretability are in fact related, as we alluded to in the introduction.

Consider the following quote, famously (but falsely) attributed to Albert Einstein:

> "If you can't explain it to a six year old, you don't understand it yourself."
>
> – Not Albert Einstein

If a network generalizes well, i.e., *understands* the data well, then it successfully filters away much irrelevant information, and the remaining signal should be low-dimensional enough for humans to make sense of. NMF successfully extracts that signal from high-dimensional NN activations.

The heatmaps derived from $\boldsymbol{U}$ suggested CNNs form a cluster hierarchy in deep feature space, with clusters representing objects that decompose into parts and further into sub-parts. In most cases, these hierarchies reflected human intuition, e.g., *person→limbs→arms*, but not always, e.g., *car→wheels→bottom of wheels*. We quantified the semantics contained in the heatmaps by performing a series co-localization and co-segmentation experiments applied to objects and object parts, and found the heatmaps compare favorably in their localization quality even when compared to more elaborate methods.

In Chapter 5 we exploited the NMF matrices to derive a global image descriptor used for content-based image search, i.e., where both the query and the results to be returned are images. Using the matrix $\boldsymbol{V}$ we performed *semantic* image search, successfully retrieving images on other instances belonging to the same semantic category. By combining information from both $\boldsymbol{V}$ and $\boldsymbol{U}$, we conducted *instance-based* image search, where the retrieved images depicted the same object instance. For the latter, we once more observed state-of-the-art retrieval performance for methods using a pre-trained CNN.

6.2 Future work

Having shown that the non-negative rank of activation matrices upper bounds the amount of memorization, a natural step is to minimize it as a form of regularization. There are, however, a number of obstacles to overcome.

First, as the non-negative rank is NP-hard to compute, so is its gradient. Minimizing it directly is therefore not possible. The proxy measure we derive using approximate NMF over a grid of different rank values K is not directly usable for this end. Although the computational pipeline that is defined by NMF with multiplicative updates *is differentiable*, its gradient properties are poor. Due to repeated matrix multiplication operations, the gradient tends to vanish.

One possibility is to instead incorporate NMF *perturbations*, i.e., $\boldsymbol{A} + [\boldsymbol{UV} - \boldsymbol{A}]_{\text{noise}}$, as a non-differentiable noise term. This form of regularization would be to NMF what the random

ablations method (Section 2.3.4) is to *dropout* [89]. However, since the noise is not independent of the data it is not clear what effect this would have.

Additionally, computing the approximate measure does entail certain overhead, which if incurred at every training iteration can substantially slow down training. A strategy that selectively applies NMF regularization, at certain iterations with certain values of K, could alleviate this problem but would involve more hyper-parameters and tuning.

The decomposition into the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ has potentially many uses that rely on image co-segmentation. One example is style transfer [35], where the NMF decomposition could be used to match different styles to different parts in a semantically-aware way. Applications could be based on applying NMF to consecutive frames of a video, for instance for object tracking.

Finally, the emergence of a distinct part through NMF does not necessarily mean its presence promotes correct classification. Similar to the study of *adversarial examples* [90], it is interesting to see how ablating or perturbing specific column-row pairs from $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively, promotes, hurts or has no effect on the correctness of network predictions.

Bibliography

[1] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *International Conference on Learning Representations (ICLR)*, 2018.

[2] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning (ICML)*, 2018.

[3] Devansh Arpit, Stanisław Jastrzkbski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. *International Conference on Machine Learning (ICML)*, 2017.

[4] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. From generic to specific deep representations for visual recognition. In *Computer Vision and Pattern Recognition (CVPR) workshop*, 2015.

[5] Artem Babenko and Victor Lempitsky. Aggregating local deep features for image retrieval. In *The IEEEConference on Computer Vision (ICCV)*, 2015.

[6] Arindam Banerjee, Inderjit S Dhillon, Joydeep Ghosh, and Suvrit Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Journal of Machine Learning Research (JMLR)*, 6(Sep):1345–1382, 2005.

[7] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[8] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 6240–6249, 2017.

[9] Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo, and Tsuhan Chen. icoseg: Interactive co-segmentation with intelligent scribble guidance. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3169–3176. IEEE, 2010.

[10] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Computer Vision and Pattern Recognition (CVPR)*, pages 3319–3327. IEEE, 2017.

[11] Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual information neural estimation. In *International Conference on Machine Learning (ICML)*, 2018.

[12] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(8):1798–1828, 2013.

[13] Mariusz Bojarski, Philip Yeres, Anna Choromanska, Krzysztof Choromanski, Bernhard Firner, Lawrence Jackel, and Urs Muller. Explaining how a deep neural network trained with end-to-end learning steers a car. *arXiv preprint arXiv:1704.07911*, 2017.

[14] Gábor Braun and Sebastian Pokutta. Common information and unique disjointness. *Algorithmica*, 76(3):597–629, 2016.

[15] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. SGD learns over-parameterized networks that provably generalize on linearly separable data. In *International Conference on Learning Representations (ICLR)*, 2018.

[16] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1721–1730. ACM, 2015.

[17] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *arXiv preprint arXiv:1611.01838*, 2016.

[18] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1971–1978, 2014.

[19] M. Cho, S. Kwak, C. Schmid, and J. Ponce. Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[20] Andrzej Cichocki and Rafal Zdunek. Multilayer nonnegative matrix factorisation. *Electronics Letters*, 42(16):1, 2006.

[21] Edo Collins and Sabine Süsstrunk. Deep feature factorization for content-based image retrieval and localization. In *IEEE International Conference on Image Processing (ICIP)*, 2019.

[22] Edo Collins, Radhakrishna Achanta, and Sabine Susstrunk. Deep feature factorization for concept discovery. In *The European Conference on Computer Vision (ECCV)*, 2018.

[23] Edo Collins, Siavash Arjomand Bigdeli, and Sabine Süsstrunk. Detecting Memorization in ReLU Networks. *arXiv preprint arXiv:1810.03372*, 2018.

[24] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

[25] Thomas Deselaers, Bogdan Alexe, and Vittorio Ferrari. Weakly supervised localization and learning with generic knowledge. *International Journal of Computer Vision (IJCV)*, 100(3):275–293, 2012.

[26] Inderjit S Dhillon, James Fan, and Yuqiang Guan. Efficient clustering of very large document collections. In *Data mining for scientific and engineering applications*, pages 357–381. Springer, 2001.

[27] Chris Ding and Xiaofeng He. K-means clustering via principal component analysis. In *International Conference on Machine Learning (ICML)*, page 29. ACM, 2004.

[28] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *34thConference on Machine Learning (ICML)*, 2017.

[29] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul): 2121–2159, 2011.

[30] Gintare Karolina Dziugaite and Daniel M Roy. Neural network matrix factorization. *arXiv preprint arXiv:1511.06443*, 2015.

[31] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

[32] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html, 2010.

[33] Samuel Fiorini, Volker Kaibel, Kanstantsin Pashkovich, and Dirk Oliver Theis. Combinatorial bounds on nonnegative rank and extended formulations. *Discrete Mathematics*, 313(1):67 – 83, 2013.

[34] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[35] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016.

[36] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 249–256, 2010.

[37] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Conference on Artificial Intelligence and statistics*, pages 315–323, 2011.

[38] Albert Gordo and Diane Larlus. Beyond instance-level image retrieval: Leveraging captions to learn a global visual representation for semantic retrieval. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6589–6598, 2017.

[39] Albert Gordo, Jon Almazán, Jerome Revaud, and Diane Larlus. Deep image retrieval: Learning global representations for image search. In *European Conference on Computer Vision (ECCV)*. Springer, 2016.

[40] Emad M Grais and Hakan Erdogan. Single channel speech music separation using nonnegative matrix factorization and spectral masks. In *Digital Signal Processing (DSP)*, pages 1–6. IEEE, 2011.

[41] David Guillamet and Jordi Vitria. Non-negative matrix factorization for face recognition. In *Topics in Artificial Intelligence*, pages 336–344. Springer, 2002.

[42] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Simultaneous detection and segmentation. *European Conference on Computer Vision (ECCV)*, 2014.

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on computer vision*, pages 1026–1034, 2015.

[44] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[45] Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

[46] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[47] Jimenez, Albert and Alvarez, Jose M., and Giro-i-Nieto, Xavier. Class-Weighted Convolutional Features for Visual Instance Search. In *British Machine Vision Conference (BMVC)*, 2017.

[48] Ian T Jolliffe. Principal component analysis and factor analysis. In *Principal component analysis*, pages 115–128. Springer, 1986.

[49] Armand Joulin, Kevin Tang, and Li Fei-Fei. Efficient image and video co-localization with frank-wolfe algorithm. In *European Conference on Computer Vision (ECCV)*, pages 253–268. Springer, 2014.

[50] Yannis Kalantidis, Clayton Mellina, and Simon Osindero. Cross-dimensional weighting for aggregated deep convolutional features. In *The European Conference on Computer Vision (ECCV)*. Springer, 2016.

[51] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *Conference on Learning Representations (ICLR)*, 2015.

[52] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Conference on Learning Representations (ICLR)*, 2015.

[53] Hartmut Klauck. Rectangle size bounds and threshold covers in communication complexity. In *Conference on Computational Complexity (CCC)*, pages 118–134. IEEE, 2003.

[54] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[55] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NeurIPS)*, pages 1097–1105, 2012.

[56] Sebastian Lapuschkin, Alexander Binder, Grégoire Montavon, Klaus-Robert Müller, and Wojciech Samek. The lrp toolbox for artificial neural networks. *Journal of Machine Learning Research*, 2016.

[57] Hieu Le, Chen-Ping Yu, Gregory Zelinsky, and Dimitris Samaras. Co-localization with category-consistent features and geodesic distance propagation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 1103–1112, 2017.

[58] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *IEEE*, 86(11):2278–2324, November 1998.

[59] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

[60] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems (NeurIPS)*, pages 556–562, 2001.

[61] Hao Li, Zheng Xu, Gavin Taylor, and Tom Goldstein. Visualizing the loss landscape of neural nets. *arXiv preprint arXiv:1712.09913*, 2017.

[62] Yao Li, Lingqiao Liu, Chunhua Shen, and Anton van den Hengel. Image co-localization by mimicking a good detector's confidence score distribution. In *European Conference on Computer Vision (ECCV)*, pages 19–34. Springer, 2016.

[63] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *IEEE Conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

[64] Dmytro Mishkin and Jiri Matas. All you need is a good init. *International Conference on Learning Representations (ICLR)*, 2016.

[65] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73, 2018.

[66] Ari S Morcos, David GT Barrett, Neil C Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. In *International Conference on Learning Representations (ICLR)*, 2018.

[67] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPS workshop on deep learning and unsupervised feature learning*, number 2, page 5, 2011.

[68] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5947–5956, 2017.

[69] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A PAC-bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

[70] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. doi: 10.23915/distill.00007. https://distill.pub/2017/feature-visualization.

[71] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[72] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition (CVPR)*, 2007.

[73] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Computer Vision and Pattern Recognition (CVPR)*, 2008.

[74] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[75] Filip Radenović, Giorgos Tolias, and Ondřej Chum. CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In *European Conference on Computer Vision (ECCV)*. Springer, 2016.

[76] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should I trust you?: Explaining the predictions of any classifier. *Conference on Knowledge Discovery and Data Mining (SIGKDD)*, pages 1135–1144, 2016.

[77] Michael Rubinstein, Armand Joulin, Johannes Kopf, and Ce Liu. Unsupervised joint object discovery and segmentation in internet images. *Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[78] Jose C Rubio, Joan Serrat, Antonio López, and Nikos Paragios. Unsupervised co-segmentation through region matching. In *Computer Vision and Pattern Recognition (CVPR)*, pages 749–756. IEEE, 2012.

[79] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533, 1986.

[80] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[81] Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3856–3866, 2017.

[82] J. Salamon, C. Jacoby, and J. P. Bello. A dataset and taxonomy for urban sound research. In *Conference on Multimedia*, pages 1041–1044, Nov. 2014.

[83] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *See https://arxiv. org/abs/1610.02391 v3*, 7(8), 2016.

[84] Ohad Shamir, Sivan Sabato, and Naftali Tishby. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29-30):2696–2711, 2010.

[85] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *Computer Vision and Pattern Recognition (CVPR) workshop*, pages 806–813, 2014.

[86] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[87] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[88] Daniel Soudry, Elad Hoffer, and Nathan Srebro. The implicit bias of gradient descent on separable data. In *International Conference on Learning Representations (ICLR)*, 2018.

[89] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, 2014.

[90] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014. URL http://arxiv.org/abs/1312.6199.

[91] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of CNN activations. In *International Conference on Learning Representations (ICLR)*, 2016.

[92] Vladimir Vapnik. *Statistical learning theory. 1998*. Wiley, New York, 1998.

[93] Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.

[94] Sara Vicente, Carsten Rother, and Vladimir Kolmogorov. Object cosegmentation. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2217–2224. IEEE, 2011.

[95] Thanh T Vu, Benjamin Bigot, and Eng Siong Chng. Combining non-negative matrix factorization and deep neural networks for speech enhancement and automatic speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP)*, pages 499–503. IEEE, 2016.

[96] Jianyu Wang and Alan L Yuille. Semantic part segmentation using compositional model combining shape and appearance. *CVPR*, 2015.

[97] Shengjie Wang, Abdel-rahman Mohamed, Rich Caruana, Jeff Bilmes, Matthai Plilipose, Matthew Richardson, Krzysztof Geras, Gregor Urban, and Ozlem Aslan. Analysis of deep neural networks with extended data jacobian matrix. In *International Conference on Machine Learning (ICML)*, pages 718–726, 2016.

[98] Aaron Wyner. The common information of two dependent random variables. *Transactions on Information Theory*, 21(2):163–179, 1975.

[99] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[100] Ge Xu, Wei Liu, and Biao Chen. Wyners common information for continuous random variables-a lossy source coding interpretation. In *Information Sciences and Systems (CISS)*, pages 1–6. IEEE, 2011.

[101] Wei Xu, Xin Liu, and Yihong Gong. Document clustering based on non-negative matrix factorization. In *Conference on Research and Development in Informaion Retrieval (SIGIR)*, pages 267–273. ACM, 2003.

[102] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *Conference on Learning Representations (ICLR)*, 2017.

[103] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929. IEEE, 2016.

**Ce document a été imprimé au Centre d'impression EPFL,
imprimerie climatiquement neutre, certifiée myClimate.**