ABSTRACT

Title of dissertation: Alternating Optimization: Constrained Problems, Adversarial Networks, and Robust Models

Zheng Xu
Doctor of Philosophy, 2019

Dissertation directed by: Professor Tom Goldstein
Department of Computer Science

Data-driven machine learning methods have achieved impressive performance for many industrial applications and academic tasks. Machine learning methods usually have two stages: training a model from large-scale samples, and inference on new samples after the model is deployed. The training of modern models relies on solving difficult optimization problems that involve nonconvex, nondifferentiable objective functions and constraints, which is sometimes slow and often requires expertise to tune hyperparameters. While inference is much faster than training, it is often not fast enough for real-time applications. We focus on machine learning problems that can be formulated as a minimax problem in training, and study alternating optimization methods served as fast, scalable, stable and automated solvers.

First, we focus on the alternating direction method of multipliers (ADMM) for constrained problem in classical convex and nonconvex optimization. Some popular machine learning applications including sparse and low-rank models, regularized

linear models, total variation image processing, semidefinite programming, and consensus distributed computing. We propose *adaptive ADMM (AADMM)*, which is a fully automated solver achieving fast practical convergence by adapting the only free parameter in ADMM. We further automate several variants of ADMM (relaxed ADMM, multi-block ADMM and consensus ADMM), and prove convergence rate guarantees that are widely applicable to variants of ADMM with changing parameters. We release the fast implementation for more than ten applications and validate the efficiency with several benchmark datasets for each application. Second, we focus on the minimax problem of generative adversarial networks (GAN). We apply prediction steps to stabilize stochastic alternating methods for the training of GANs, and demonstrate advantages of GAN-based losses for image processing tasks. We also propose GAN-based knowledge distillation methods to train small neural networks for inference acceleration, and empirically study the trade-off between acceleration and accuracy. Third, we present preliminary results on adversarial training for robust models. We study fast algorithms for the attack and defense for universal perturbations, and then explore network architectures to boost robustness.

Alternating Optimization: Constrained Problems, Adversarial Networks, and Robust Models

by

Zheng Xu

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Advisory Committee:
Professor Tom Goldstein, Chair/Advisor
Professor Rama Chellappa
Professor David Jacobs
Professor Gavin Taylor
Professor Furong Huang
Professor John Dickerson

Dedication

To my parents JX, MS, and my wife XL.

Acknowledgments

Now I have (almost) reached the end of the Ph.D. journal. I owe a debt of gratitude to many people, and I apologize in advance to whom I could have inadvertently left out in this acknowledgement.

First and foremost I would like to thank my advisor, Tom Goldstein, who has provided everything I could ask for from an advisor. I have learnt so much by working together, talking with him, and simply being around him. His views on sharing credit, preferring simplicity, driven by intuition and motivation, seeking long-term impact, prioritization, and presentation have made a huge impact on and will continue guiding my own research. He gave me a lot of freedom to exploring my research, taking internships, and spending time with my family. It has been a great pleasure working with such a nice and intelligent person.

I thank my committee members, David Jacobs, Gavin Taylor, Furong Huang, Rama Chellappa, and John Dickerson. In our collaboration, they have helped a lot with their expertise in computer vision, tensor method, distributed computing, etc. This thesis would not become the current shape without these fruitful collaborations and their valuable feedback and suggestions. I have taken several research internships and I thank my industrial mentors, Doug Burdick at IBM, Anima Anandkumar at Amazon, Jiawei Huang at Honda, Chen Fang at Adobe, and Oncel Tuzel at Apple, for giving me the opportunities to touch the industrial research style, enjoy the computing resources, and stay in bay area. Part of the thesis is based on my internship projects at Honda and Adobe. I would like to thank Louiqa

Raschid from UMD business school for her financial support at the beginning of my PhD study. I enjoyed the financial projects we have worked on together and appreciate the freedom she gave me for exploring my interests. Thank my mentors and teachers before my Ph.D. study, my undergraduate advisor Houqiang Li and my master advisor Chang-Wen Chen, for their guidance and recommendation in my early research stage. Particularly, I would also like to thank my mentor Xin-Jing Wang at Microsoft during my master study, for her strong support in my application at the beginning and job search at the end of my Ph.D. study.

I have collaborated with many fantastic researchers during my Ph.D. study. Special thanks go to Mario Figueiredo for helping organize and polish my ADMM papers, I would hope I could write like him one day; Xiaoming Yuan for his theoretical insights that lead to the analysis in my paper; Hao Li for helping my transition from classical optimization to neural networks and as a nice hardware and system consult; Ali Shafahi for our collaboration on adversarial attack and defense; Xitong Yang for our discussion on image processing tasks. Besides my committee members and industrial mentors, I am also fortune to work with our long-term collaborator Christoph Studer from Cornell; our labmates Soham De, Sohil Shah, Abhay Yadav, Ronny Huang, Mahyar Najibi and Karthik Sankararaman from UMD; Yen-Chang Hsu from Gatech, Michael Wilber from Cornell Tech, Aaron Hertzman, Hailin Jin from Adobe during my internships; my old friends Kuiyuan Yang from DeepMotion, Xiaoshuai Sun from HIT, Wen Li from ETH, and many other researchers.

I have met new and old friends in this journey. Hao Zhou and Ruofei Du have helped my quick adaptation to life in Maryland. Jiayao Hu has always been

welcoming me and my family to California. I did a lot of my course projects with Weiwei Yang and Hong Wei. My five-year roommate Peng Lei has shared a lot of his experience. I enjoyed all the talk, discussion and play at Maryland, during my internships and at conferences with my friends, Ang Li, Xing Niu, Jinfeng Rao, Xiyang Dai, Han Zhou, Changqing Zou, Mingfei Gao, Jingjing Zheng, Chen Zhu, Renkun Ni, Lingjia Deng, Beidi Chen, Yang Shi, Tan Yu, Hui Ding, Yin Zhou, Shuangfei Zhai, Wei Ping, Ye Xu, Amin Ghiasi, to name a few. I would like to thank the staff members who has created the cheerful environment at the computer science department, Jeff Foster, Jeniffer Story, Tom Hurst, Jodie Gray, Sharron McElroy and others.

Last but most importantly, I thank my family, my parents Jinxin Xu and Mingxia Sun, for their unconditional love, understanding and support. I can keep exploring and enjoying a researcher's life as I know they always have my back. I am extremely fortune to have the experience together with my wife Xue Li, who deserves more credit then myself. She has made me a better researcher and a better person. If there is one thing that I did not do wrong and would not regret at all in the past five years, it is sharing the adventure with her.

So long, and thanks for all the fish.

Table of Contents

List of Tables

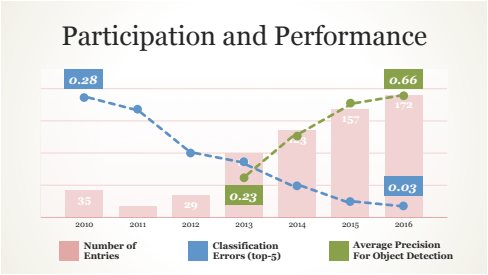
List of Figures

Chapter 1: Introduction

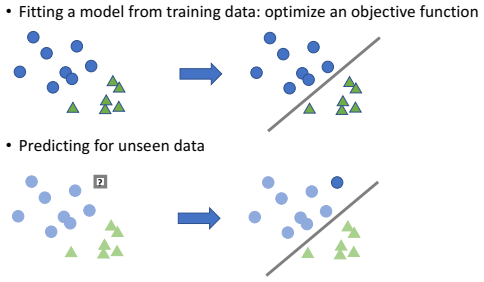
Data-driven machine learning methods have become a success in both industrial applications and academic research. For example, the best performed model has surpassed human performance in imagenet recognition competition, which is a challenge of classifying 1000-categories for more than 10 millions of images, see Fig. 1.1a for recent trend of this competition. The recent advance of machine learning benefits from growing available data on the web and supervised information from crowd sourcing, powerful computing devices like GPUs, and strong models like deep neural networks to fit the data. Large scale data and complicated models also bring challenges for machine learning.

Machine learning methods usually have two stages: training a model from large-scale samples, and inference on new samples after the model is deployed, see an illustrative example in Fig. 1.1b. The training of modern models relies on solving difficult optimization problems that involve nonconvex, nondifferentiable objective functions and constraints, which is sometimes slow and often requires expertise to tune hyperparameters. Fig. 1.2 presents examples of difficult problems to be optimized with complex solvers. While inference is much faster than training, it is often not fast enough for real-time in practice. How to efficiently fit large-scale

data and deploy models in domain-specific applications is one of the key problems in machine learning.



(a) Machine learning models surpass human error rate of 0.05 on large scale image classification task. [FFD17]



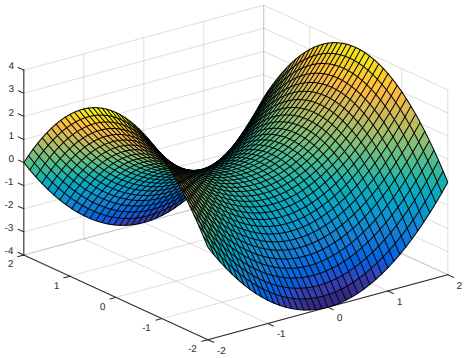
(b) Training and inference stages of machine learning method.

Figure 1.1: The success of data-driven mathine learning.

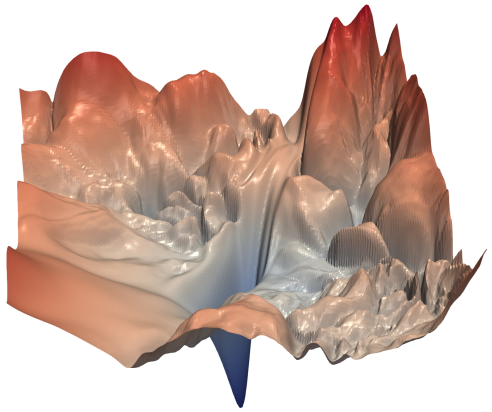
We focus on machine learning problems that can be formulated as a minimax problem in training, and study alternating optimization methods served as fast, scalable, stable and automated solvers. Our study mainly concentrates on classification and image processing tasks, including the applications of both classical linear models with regularizers and modern deep neural networks. We proposed an automated solver for constrained problem that converges fast in practice. Next, we stabalize the training of adversarial networks and apply the adversarial loss for image processing and network acceleration. Finally, we develop fast algorithm for adversarial attack and defense and exploit network design for robustness.

1.1 Organization

Alternating optimization methods optimize complex objective functions by solving simpler sub-problems or related steps. Some popular applications including sparse and low-rank models, regularized linear models, total variation image pro-



(a) Finding a saddle point of a surface requires complex solvers such as alternating optimization methods.



(b) The loss surface of a real neural network that is often optimized by simple stochastic gradient methods; the behavior of these solvers is not well understood.

Figure 1.2: Examples show the difficulty of optimization in machine learning.

cessing, generative adversarial networks, and adversarial training for robust models [Xu+14; Xu+15; Xu+17b; Xu+16b; Xu+16a; Li+17d; Yad+18; Sha+19].

In the first part of the dissertation (Chapter 2 - Chapter 5), we introduce our automated solver for constrained problem based on our series of works that make many machine learning problems easier for non-expert users [Xu+15; Xu+17b; Xu+17d; Xu+17c; Xu+17e; Xu+16a]. Among the alternating optimization methods, the alternating direction method of multipliers (ADMM) is a versatile tool for solving a wide range of problems that can be formulated as a two-term objective with a linear constraint. ADMM applies three steps to solve the saddle point of Lagrangian minimax problem . The solver has the only free parameter, known as penalty parameter in the literature. We proposed *adaptive ADMM (AADMM)*, which is a fully automated solver that tunes the penalty parameter in an adaptive way. We released code for more than ten applications such as sparse linear

regression, support vector machine classifiers, semidefinite programming, and image denoising [Xu+17b; Xu+16a]. By investigating the equivalence of primal and dual forms of ADMM, we derived an adaptive penalty parameter schema inspired by the Barzilai-Borwein methods in gradient descent. We also propose an intuitive while efficient rule to safeguard our parameter estimation to guarantee convergence. The overhead of AADMM over ADMM is modest–only a few inner products plus the storage to keep one previous iterate, while the practical convergence speed of AADMM is much faster than non-adaptive methods. We also prove convergence rate guarantees that are widely applicable to variants of ADMM with changing penalty parameter [Xu+17c]. With a bounded adaptivity assumption, we prove a worst case ergodic $O(1/k)$ convergence rate in variational inequality sense.

We then automate variants of ADMM by exploiting the proposed adaptive schema. A practical variant of ADMM is adding an extra interpolation step between the two primal update steps, which also introduces a new parameter, the relaxation parameter. We proposed *adaptive relaxed ADMM (ARADMM)* to jointly choose the penalty and relaxation parameters, which achieves even faster practical convergence than AADMM for some applications, and is also guaranteed to converge [Xu+17d]. ADMM methods, which typically deal with two-term objectives, can be generalized to the "multi-block" ADMM, which handles many. Though the convergence analysis becomes significantly harder and it sometimes cannot converge in theory, we proposed *adaptive multi-block ADMM (AMADMM)* that works well in practice [Xu+17e]. ADMM is also one of the most efficient methods for distributed optimization [Boy+11]. We proposed a variant of ADMM that is scalable

in distributing the training of deep neural networks [Tay+16; Tay+17], although it still suffers from the generalization issue that is common in batch training for very deep nets. We proposed *adaptive consensus ADMM (ACADMM)* for choosing node-specific penalty parameters and developed a fully automated solver for distributing general optimization problems [Xu+17c].

In the second part of the dissertation (Chapter 6 - Chapter 8), we discuss generative adversarial networks (GANs) and various applications [Yad+18; Yan+18; Xu+18b; Xu+18a; Xu+19a]. The training of GANs is a minimax problem relies heavily on stochastic alternating direction methods as the solver, and there is increasing interests in optimizing such nonconvex and nonconcave objectives. The training of GANs is not well understood and is also known to be unstable. We showed that the stochastic alternating methods for training GANs can be stabilized by introducing the prediction step inspired by the primal-dual gradient methods, which allows users to choose bigger stepsizes and achieves better models [Yad+18]. We analyze the convergence of stochastic alternating methods with prediction step for convex-concave problem in Chapter 6 to help understand the solver in theory.

GANs have been extensively studied over recent years, especially for image processing tasks. We recently applied GANs for image style transfer [Xu+19a] and image dehazing [Xu+18a; Yan+18]. We use the encoder-decoder architecture and pre-trained feature extractor for these image processing tasks. Together with normalization layers, we can train deeper and wider networks that achieve superior performance. Deep and wide networks can sometimes be difficult to deploy in practice. We further show that GAN framework can be applied when we learn a small

student network from a large teacher network for accelerating inference. Our results suggest GAN framework can not only boost the performance of image processing, but also apply for many other tasks [Xu+18b].

In the third part of the dissertation (Chapter 9 - Chapter 10), we study robust models that can defend against adversarial examples. Adversarial examples can be generated by adding a small perturbation to test samples (often created by first-order gradient method for attacking the model) that mislead models to make mistakes on prediction. It is more difficult to attack a robust model, and adversarial training is one of the most successful methods for defense. Adversarial training can be formulated as a minimax problem of training on generated adversarial examples. In Chapter 9 [Sha+18], we focus on universal perturbation that can fool a model when applied to a set of samples. Simple methods like stochastic gradient and clipping loss help us efficiently generate universal perturbation for attack, and make the attack and defense of universal perturbation on large scale dataset like ImageNet possible. Our adversarially trained model is robust to universal perturbations. In Chapter 10 [Xu+19b], we present preliminary results on exploiting network architecture to boost robustness. We show adaptive normalization layer can be helpful for training robust models. Chapter 11 concludes the dissertation with some discussion remarks.

1.2 Contribution

We summarize the technical contributions here:

- For the first time, we prove $O(1/k)$ convergence rate of ADMM with adaptive penalty parameter. The assumption can be satisfied in practice and the theoretical analysis is widely applicable to the variants of ADMM (Chapter 3).

- We proposed Adaptive ADMM (AADMM), which is a fully automated solver that converges fast in practice. It is easy for non-expert users to use our AADMM for their domain-specific problems. We design spectral stepsize for constrained problem, which is simple yet effective. We validated the performance of AADMM with various applications and benchmark datasets (Chapter 4) .

- We further exploit the proposed adaptive schema for variants of ADMM. Adaptive relaxed ADMM achieves even faster practical convergence. Adaptive consensus ADMM is more suitable for distributed computing withe large number of nodes. The adaptive schema can be extended to multi-block ADMM, and AADMM is effective for several nonconvex problems. We perform extensive experimental study and open-sourced our fast implementation (Chapter 5).

- For a convex-concave saddle point problem, we show that stochastic alternating gradients may not converge, while introducing the extra prediction step would help it converge. The stochastic alternating gradients with predictions step has $O(1/\sqrt{k})$ convergence rate (Chapter 6).

- In Chapter 7, we apply adversarial networks to enhance image processing. For image style transfer, we adversarially training a single feed-forward network to

learn from multi-domain artistic images for arbitrary style transfer. For image dehazing, we use a simple yet efficient network that can serve as a hard to beat baseline, and apply GAN framework to train without paired image data.

- We apply adversarial network beyond image processing tasks. We introduce conditional adversarial networks to transfer knowledge from large teacher network to small student student for inference acceleration. We empirically show that the loss learned by the adversarial training has the advantage over the predetermined loss in the student-teacher strategy, especially when the student network has relatively small capacity (Chapter 8).

- We propose fast algorithm based on stochastic gradients for the attack and defense of universal perturbation. Our fast algorithm enables us to generate stronger universal perturbation, and train robust models for large scale dataset like ImageNet (Chapter 9).

- In Chapter 10, We propose to adversarially train adaptive networks for robustness. To build adaptive networks, we introduce a normalization module conditioned on inputs which allows the network to "adapt" itself for different samples. We found increasing the stepsize for generating adversarial examples and initializing from pre-trained natural model can help adversarial training.

Part I

Constrained Problem and Adaptive ADMM

Chapter 2: Constrained Problem and ADMM

Alternating optimization methods are widely used to solve problems in machine learning, computer vision, and image processing. Some popular applications including sparse and low-rank models, empirical risk minimization, total variation image restoration, and generative adversarial networks [Boy+11; Gol+14b; Goo+14a; Xu+17b; Li+17d]. In this section, we first review ADMM methods and its variants, and introduce background on general minimax optimization methods and the stochastic alternating methods. We then provide multiple applications in computer vision, image processing and machine learning. At last, we discuss previous theoretical analysis on the convergence of these methods. In the next chapter, we will prove convergence for ADMM with adaptive parameters.

2.1 ADMM and penalty parameter

ADMM dates back to the 1970s [GM76; GM75]. In the last decade, ADMM became one of the tools of choice to handle a wide variety of optimization problems in machine learning, signal processing, and many other areas (for a comprehensive

review, see [Boy+11]). ADMM tackles problems in the form

$$\min_{u,v} H(u) + G(v), \quad \text{subject to} \quad Au + Bv = b. \tag{2.1}$$

We optimize the Lagrangian of the constrained problem

$$\max_{\lambda} \min_{u,v} H(u) + G(v) + \langle \lambda, b - Au - Bv \rangle + \frac{\tau}{2} \|b - Au - Bv\|^2 \tag{2.2}$$

with three steps per iteration as

$$u_{k+1} = \arg\min_{u} H(u) + \frac{\tau_k}{2} \|b - Au - Bv_k + \frac{\lambda_k}{\tau_k}\|_2^2 \tag{2.3}$$

$$v_{k+1} = \arg\min_{v} G(v) + \frac{\tau_k}{2} \|b - Au_{k+1} - Bv + \frac{\lambda_k}{\tau_k}\|_2^2 \tag{2.4}$$

$$\lambda_{k+1} = \lambda_k + \tau_k (b - Au_{k+1} - Bv_{k+1}), \tag{2.5}$$

where λ is the dual variable. The penalty parameter $\{\tau_k\}$ is the only free parameter in ADMM and is significant for the practical convergence speed.

2.1.1 Residuals and stop condition

The convergence of the algorithm can be monitored using primal and dual "residuals," both of which approach zero as the iterates become more accurate, and which are defined as

$$r_k = b - Au_k - Bv_k, \qquad \text{and} \qquad d_k = \tau_k A^T B(v_k - v^{k-1}), \tag{2.6}$$

respectively [Boy+11]. The iteration is generally stopped when

$$\|r_k\|_2 \leq \epsilon^{tol} \max\{\|Au_k\|_2, \|Bv_k\|_2, \|b\|_2\} \quad \text{and} \quad \|d_k\|_2 \leq \epsilon^{tol}\|A^T\lambda_k\|_2, \tag{2.7}$$

where $\epsilon^{tol} > 0$ is the stopping tolerance.

2.2 Multi-block ADMM

Multi-block ADMM is the direct extension of two-block ADMM, which tackles problems in the form

$$\min_{u_i \in \mathbb{R}^{n_i}} \sum_{i=1}^{N} H_i(u_i), \quad \text{subject to} \quad \sum_{i=1}^{N} A_i u_i = b, \tag{2.8}$$

where N is the number of blocks, $H_i : \mathbb{R}^{n_i} \to \bar{\mathbb{R}}$ are (convex) functions, $A_i \in \mathbb{R}^{p \times n_i}$, and $b \in \mathbb{R}^p$. With $\lambda \in \mathbb{R}^p$ denoting the dual variables (Lagrange multipliers), multi-block ADMM has the form

$$\begin{aligned} u_{i,k+1} = &\arg\min_{u_i} H_i(u_i) + \langle \lambda_k, -A_i u_i \rangle \\ &+ \frac{\tau_k}{2}\|b - \sum_{j=1}^{i-1} A_j u_{j,k+1} - A_i u_i - \sum_{j=i+1}^{N} A_j u_{j,k}\|_2^2 \end{aligned} \tag{2.9}$$

$$\lambda_{k+1} = \lambda_k + \tau_k(b - \sum_{i=1}^{N} A_i u_{i,k+1}), \tag{2.10}$$

where (2.9) represents the sequential update of primal variables u_i in the order of $i = 1, \ldots, N$. And (2.9) recovers the vanilla ADMM steps in (5.157, 5.158) when $N = 2$.

2.2.1 Residuals and stop condition

The primal and dual residuals are defined to measure the primal and dual feasibility of the constrained problem. Following the derivatioin of two-block ADMM in [Boy+11], the primal and dual feasibility of multi-block ADMM are

$$0 = b - \sum_{i=1}^{N} A_i u_i \tag{2.11}$$

$$0 \in \partial H_i(u_i) - A_i^T \lambda. \tag{2.12}$$

The primal residual is defined as

$$r_{k+1} := b - \sum_{i=1}^{N} A_{i,k+1} u_{i,k+1}. \tag{2.13}$$

From multi-block ADMM steps (2.9,2.10),

$$0 \in \partial H_i(u_{i,k+1}) - A_i^T \lambda_k - \tau_k A_i^T (b - \sum_{j=1}^{i} A_j u_{j,k+1} - \sum_{j=i+1}^{N} A_j u_{j,k}) \tag{2.14}$$

$$= \partial H_i(u_{i,k+1}) - A_i^T \lambda_{k+1} + \tau A_i^T \sum_{j=i+1}^{N} A_j (u_{i,k} - u_{i,k+1}). \tag{2.15}$$

$u_{N,k+1}, \lambda_{k+1}$ always satisfy the dual feasibility condition in (2.12). Dual residual are defined to measure the other $N-1$ conditions

$$d_{i,k+1} := \tau A_i^T \sum_{j=i+1}^{N} A_j (u_{i,k} - u_{i,k+1}). \tag{2.16}$$

We define stop criterion based on multi-block residuals as

$$\begin{cases} \|r_k\|_2 & \leq \epsilon^{tol} \max\{\max_i\{\|A_i u_{i,k}\|_2\}, \|b\|_2\} \\ \max_i\{\|d_{i,k}\|_2\} & \leq \epsilon^{tol} \min_i\{\|A_i^T \lambda_k\|_2\}, \end{cases} \tag{2.17}$$

where $\epsilon^{tol} > 0$ is the stopping tolerance.

2.3 Minimax optimization problems

Let us consider optimizing the minimax problem that is more general than the Lagrangian saddle point problem (2.2),

$$\min_u \max_v \mathcal{L}(u, v). \tag{2.18}$$

An attractive application of the minimax problem is the training of generative adversarial networks (GANs). The stochastic alternating gradients method for training GANs can be written as

$$u_{k+1} = u^k - \tau_k \mathcal{L}'_u(u^k, v^k) \quad | \quad \text{gradient descent in } u \tag{2.19}$$

$$v_{k+1} = v^k + \sigma_k \mathcal{L}'_v(\bar{u}_{k+1}, v^k) \quad | \quad \text{gradient ascent in } v, \tag{2.20}$$

where $\{\tau_k, \sigma_k\}$ are the stepsizes.

When $\mathcal{L}(u, v)$ is convex in u, and concave in v, we are particularly interested

in the following bilinear saddle point problem,

$$\min_u \max_v \mathcal{L}(u, v) = F(u) + v^T A u - G(v). \tag{2.21}$$

The primal-dual hybrid gradients method (PDHG) for this problem has the following steps,

$$\hat{u}_{k+1} = u_k - \tau_k A^T v_k \tag{2.22}$$

$$u_{k+1} = \arg\min_u F(u) + \frac{1}{2\tau_k}\|u - \hat{u}_{k+1}\|_2^2 \tag{2.23}$$

$$\tilde{u}_{k+1} = u_{k+1} + (u_{k+1} - u_k) \tag{2.24}$$

$$\hat{v}_{k+1} = v_k + \sigma_k A \tilde{v}_{k+1} \tag{2.25}$$

$$v_{k+1} = \arg\min_v G(v) + \frac{1}{2\sigma_k}\|v - \hat{v}_{k+1}\|^2, \tag{2.26}$$

where $F(u), G(v)$ can be nondifferentiable functions. The convex-concave saddle point problem is understood better in theory than the non-convex and non-concave GANs. PDHG can be considered as a preconditioned ADMM [CP11].

2.3.1 Residuals and stop condition

We can use the norm of gradients $\mathcal{L}'_u, \mathcal{L}'_v$ to monitor the alternating gradient methods. In practice, when training large scale problems such as GANs in stochastic setting, we usually stop the algorithm after a fixed number of iterations.

For PDHG, the optimal condition of steps (2.23, 2.26) are

$$0 \in \partial F(u_{k+1}) + \frac{1}{\tau_k}(u_{k+1} - (u_k - \tau_k A^T v_k)) \tag{2.27}$$

$$= \partial F(u_{k+1}) + \frac{1}{\tau_k}(u_{k+1} - u_k) + A^T v_k \tag{2.28}$$

$$0 \in \partial G(v_{k+1}) + \frac{1}{\sigma_k}(v_{k+1} - (v_k + \sigma_k A(2u_{k+1} - u_k))) \tag{2.29}$$

$$= \partial G(v_{k+1}) + \frac{1}{\sigma_k}(v_{k+1} - v_k) - A(2u_{k+1} - u_k). \tag{2.30}$$

The saddle point $(u^\star, v^\star)$ satisfies

$$0 \in \partial F(u^\star) + A^T v^\star \tag{2.31}$$

$$0 \in \partial G(v^\star) - A u^\star. \tag{2.32}$$

The primal and dual residuals are defined as

$$P(u_{k+1}, v_{k+1}) = A^T(v_{k+1} - v_k) - \frac{1}{\tau_k}(u_{k+1} - u_k) \in \partial F(u_{k+1}) + A^T v_{k+1} \tag{2.33}$$

$$D(u_{k+1}, v_{k+1}) = A(u_{k+1} - u_k) - \frac{1}{\sigma_k}(v_{k+1} - v_k) \in \partial G(v_{k+1}) - A u_{k+1}. \tag{2.34}$$

2.4 Exemplar applications

2.4.1 Elastic net regularized linear regression

Elastic net (EN)is a modification of ℓ_1-regularized linear regression (a.k.a. LASSO) that helps preserve groups of highly correlated variables [ZH05; Gol+14b]

and requires solving

$$\min_x \frac{1}{2}\|Dx - c\|_2^2 + \rho_1\|x\|_1 + \frac{\rho_2}{2}\|x\|_2^2, \tag{2.35}$$

where, as usual, $\|\cdot\|_1$ and $\|\cdot\|_2$ denote the ℓ_1 and ℓ_2 norms, D is a data matrix, c contains measurements, and x is the vector of regression coefficients. One way to apply ADMM to this problem is to rewrite it as

$$\begin{aligned} &\min_{u,v} \frac{1}{2}\|Du - c\|_2^2 + \rho_1\|v\|_1 + \frac{\rho_2}{2}\|v\|_2^2 \\ &\text{subject to} \quad u - v = 0. \end{aligned} \tag{2.36}$$

The synthetic dataset introduced by Zou and Hastie [ZH05] and realistic dataset introduced by Efron et al. [Efr+04] and Zou and Hastie [ZH05] are investigated. Typical parameters are $\rho_1 = \rho_2 = 1$.

2.4.2 Low rank least squares

Low rank least squares (LRLS) uses the nuclear matrix norm (sum of singular values) as the convex surrogate of matrix rank,

$$\min_X \frac{1}{2}\|DX - C\|_F^2 + \rho_1\|X\|_* + \frac{\rho_2}{2}\|X\|_F^2, \tag{2.37}$$

where $\|\cdot\|_*$ denotes the nuclear norm, $\|\cdot\|_F$ is the Frobenius norm, $D \in \mathbb{R}^{n\times m}$ is a data matrix, $C \in \mathbb{R}^{n\times d}$ contains measurements, and $X \in \mathbb{R}^{m\times d}$ is the variable matrix. ADMM can be applied after rewriting (2.37) as Yang and Yuan [YY13] and

Xu et al. [Xu+15]

$$\begin{aligned} &\min_{U,V} \frac{1}{2}\|DU - C\|_F^2 + \rho_1\|V\|_* + \frac{\rho_2}{2}\|V\|_F^2, \\ &\text{subject to} \quad U - V = 0. \end{aligned} \tag{2.38}$$

A synthetic problem is constructed using a random data matrix $D \in \mathbb{R}^{1000\times 200}$, a low rank matrix $X \in \mathbb{R}^{200\times 500}$, and $C = DW + \text{Noise}$. We use the binary classification problems introduced by Lee et al. [Lee+06] and Schmidt et al. [Sch+07], where each column of X represents a linear exemplar classifier, trained with a positive sample and all negative samples [Xu+15]; typical parameters are $\rho_1 = \rho_2 = 1$.

2.4.3 Support vector machine and quadratic programming

The dual of the support vector machine (SVM) learning problem is a quadratic programming (QP) problem,

$$\begin{aligned} &\min_{z} \frac{1}{2} z^T Q z - e^T z \\ &\text{subject to} \quad c^T z = 0 \ \text{ and } \ 0 \le z \le C, \end{aligned} \tag{2.39}$$

where z is the SVM dual variable, Q is the kernel matrix, c is a vector of labels, e is a vector of ones, and $C > 0$ [CL11]. We also consider the canonical QP

$$\min_{x} \frac{1}{2} x^T Q x + q^T x \quad \text{subject to} \quad Dx \le c, \tag{2.40}$$

which can be solved by applying ADMM to

$$\begin{aligned} \min_{u,v} \ & \frac{1}{2} u^T Q u + q^T u + \iota_{\{z:\, z_i \leq c\}}(v) \\ \text{subject to} \quad & Du - v = 0; \end{aligned} \tag{2.41}$$

here, ι_S is the indicator function of set S: $\iota_S(v) = 0$, if $v \in S$, and $\iota_S(v) = \infty$, otherwise.

We study classification problems from Lee et al. [Lee+06] and Schmidt et al. [Sch+07] with typical parameter$C = 1$, and a random synthetic QP [Gol+14b], where $Q \in \mathbb{R}^{500\times 500}$ with condition number $\simeq 4.5 \times 10^5$.

2.4.4 Basis pursuit

Basis pursuit (BP) seeks a sparse representation of a vector c by solving the constrained problem

$$\min_x \|x\|_1 \quad \text{subject to} \quad Dx = c, \tag{2.42}$$

where $D \in \mathbb{R}^{m\times n}, c \in \mathbb{R}^m, m < n$. An extended form with $\hat{D} = [D, I] \in \mathbb{R}^{m\times(n+m)}$ has been used to reconstruct occluded and corrupted faces [Wri+09a]. To apply ADMM, problem (2.42) is rewritten as

$$\min_{u,v} \iota_{\{z:\, Dz=c\}}(u) + \|v\|_1 \quad \text{subject to} \quad u - v = 0. \tag{2.43}$$

We experiment with synthetic random $D \in \mathbb{R}^{10\times 30}$. We also use a data matrix for face reconstruction from the Extended Yale B Face dataset [Wri+09c], where each frontal face image is scaled to 32×32. For each human subject, an image is selected and corrupted with 5% noisy pixels, and the remaining images from the same subject are used to reconstruct the corrupted image.

2.4.5 Consensus ℓ_1-regularized logistic regression

Consensus ℓ_1-regularized logistic regression is formulated as a distribute optimization problem with the form

$$\begin{aligned} \min_{x_i,z} \sum_{i=1}^{N} \sum_{j=1}^{n_i} \log(1+\exp(-c_j D_j^T x_i)) + \rho\|z\|_1 \\ \text{subject to} \quad x_i - z = 0, i = 1,\ldots,N, \end{aligned} \tag{2.44}$$

where $x_i \in \mathbb{R}^m$ represents the local variable on the ith distributed node, z is the global variable, n_i is the number of samples in the ith block, $D_j \in \mathbb{R}^m$ is the jth sample, and $c_j \in \{-1, 1\}$ is the corresponding label. The goal of this example is to test AADMM also in distributed/consensus problems, for which ADMM has become an important tool [Boy+11].

A synthetic problem is constructed with Gaussian random data and sparse ground truth solutions. Binary classification problems from Lee et al. [Lee+06] and Liu et al. [Liu+09], and Schmidt et al. [Sch+07] are also used to test the effectiveness of the proposed method. We use $\rho = 1$, for small and medium datasets, and $\rho = 5$ for the large datasets to encourage sparsity. We split the data equally into two blocks

and use a loop to simulate the distributed computing of consensus subproblems.

2.4.6 Semidefinite programming

Semidefinite programming (SDP) solves the problem

$$\min_X \langle F, X \rangle \quad \text{subject to} \quad X \succeq 0, \quad \mathcal{D}(X) = c, \tag{2.45}$$

where $X \succeq 0$ means that X is positive semidefinite, $X, F, D_i \in \mathbb{R}^{n \times n}$ are symmetric matrices, inner product $\langle X, Y \rangle = \text{trace}(X^T Y)$, and $\mathcal{D}(X) = (\langle D_1, X \rangle, \ldots, \langle D_m, X \rangle)^T$. ADMM is applied to the dual form of (2.45),

$$\min_{y,S} \quad -c^T y \quad \text{subject to} \quad \mathcal{D}^*(y) + S = F, \quad S \succeq 0, \tag{2.46}$$

where $\mathcal{D}^*(y) = \sum_{i=1}^m y_i D_i$, and S is a symmetric positive semidefinite matrix.

As test data, we use 6 graphs from the *Seventh DIMACS Implementation Challenge on Semidefinite and Related Optimization Problems* (following Burer and Monteiro [BM03]).

2.4.7 Unwrapped SVM

The unwrapped formulation of SVM [Gol+16], which can be used in distributed computing environments via "transpose reduction" tricks, applies ADMM

to the primal form of SVM to solve

$$\min_x \frac{1}{2}\|x\|_2^2 + C\sum_{j=1}^{n} \max\{1 - c_j D_j^T x,\, 0\}, \tag{2.47}$$

where $D_j \in \mathbb{R}^m$ is the jth sample of training data, and $c_j \in \{-1, 1\}$ is the corresponding label. ADMM is applied by splitting the ℓ_2-norm regularizer and the non-differentiable hinge loss term.

2.4.8 Total variation image denoising

Total variation image denoising (TVID) is often performed by solving [Rud+92]

$$\min_x \frac{1}{2}\|x - c\|_2^2 + \rho\|\nabla x\|_1 \tag{2.48}$$

where c represents given noisy image, and ∇ is the discrete gradient operator, which computes differences between adjacent image pixels. ADMM is applied by splitting the ℓ_2-norm term and the non-differentiable total variation term.

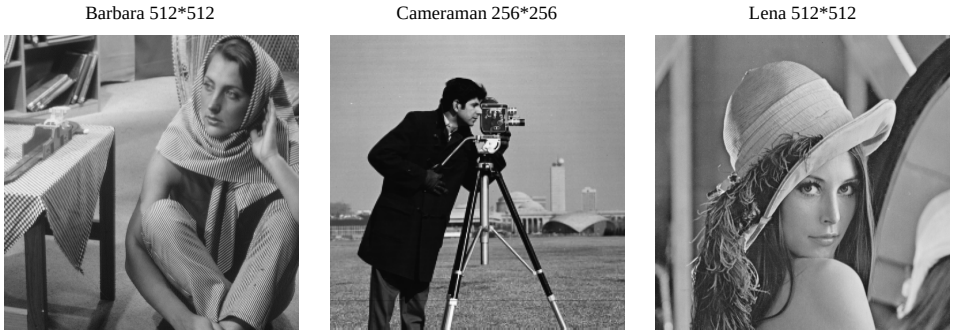


Figure 2.1: "Barbara", "Cameraman", and "Lena" for image processing applications.

2.5 Benchmark datasets

Both synthetic and benchmark datasets are discussed for those applications. The regression and classification benchmark datasets (obtained from the UCI repository [1] and the LIBSVM page [2]) are used in [Efr+04; Lee+06; Liu+09; Sch+07; ZH05]. The dataset statistics are summarized in Table 2.1. The features are centered to have zero mean and unit variance for the small and medium sized datasets, and scaled to be between -1 and 1 for the large and sparse datasets. Extended Yale Face Dataset B (obtained from [3]) with cropped and resized images (32*32) is used for basis pursuit [Wri+09c] and robust principal component analysis [Wri+09b]. This dataset has 38 individuals and around 64 near frontal images under different illuminations per individual. "Barbara", "Cameraman", and "Lena" (Fig. 2.1) are used for image processing applications.

2.6 Convergence and related work

ADMM is known to have a $O(1/k)$ convergence rate under mild conditions for convex problems [HY12b; HY15], while a $O(1/k^2)$ rate is possible when at least one of the functions is strongly convex or smooth [Gol+13; Gol+14b; Kad+15; TY16b]. Linear convergence can be achieved with strong convexity assumptions [DY14; Nis+15; TZ15; DY16; GB16]. All of these results assume constant penalty.

[1] `http://archive.ics.uci.edu/ml/`
[2] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`
[3] `http://www.cad.zju.edu.cn/home/dengcai/Data/FaceData.html`
[4] Australian Credit
[5] Australian Heart
[6] Wisconsin Breast Caner

Application	Dataset	#samples	#features
Regression	Boston	506	13
	Diabetes	768	8
	Leukemia	38	7129
	Prostate	97	8
	Servo	130	4
Binary Classification	AC[4]	690	14
	AH[5]	270	13
	German	1000	24
	Hepatitis	155	19
	Ionosphere	351	34
	Madelon	2000	500
	Pima	768	8
	Sonar	208	60
	Spambase	4601	57
	Spect	80	22
	Spectf	80	44
	Splice	1000	60
	WBC[6]	683	10
	News20	19996	1355191
	Rcv1	20242	47236
	Realsim	72309	20958

Table 2.1: Statistics of regression and classification benchmark datasets.

When the penalty is adapted, convergence is proved without providing a rate [He+00], and for some particular variants of ADMM ("linearized" or "preconditioned") [Lin+11; Gol+15] . We prove the $O(1/k)$ convergence rate for ADMM with adaptive penalty, under the assumption of convex functions and bounded adaptivity in Chapter 3.

ADMM steps can also be applied to nonconvex problems such as ℓ_0 regularized problem [DZ13], and phase retrieval [Wen+12]. The convergence of ADMM for nonconvex problems under certain assumptions are studied in [Wan+14; LP15; Hon+16; Wan+15]. The current weakest assumptions are given in [Wan+15], which requires the Lipschitz differentiable of the last updated primal term and several other conditions.

In practice, ADMM and its variants have been applied to multi-block problems such as matrix factorization [Xu+12; SF14], robust principal component analysis [TY11] and neural networks [Tay+16]. However, the direct extension of ADMM for multi-block problems is not necessarily convergent [Che+16a]. The convergence of multi-block ADMM is also discussed in [HY12a; Cai+14; Lin+15; TY16a], where at least one of the functions is strongly convex is assumed. Linear convergence is provided for multi-block ADMM in [HL12], with extra assumptions on the function form of each term, compact feasible sets and rank of matrices in linear conditions.

The primal-dual hybrid gradient (PDHG) method [ZC08; Ess+09] has been popularized by Chambolle and Pock [CP11], and has been successfully applied to a range of machine learning and statistical estimation problems [Gol+15]. Stochastic methods for convex saddle-point problems can be roughly divided into two cate-

gories: stochastic coordinate descent [DL14; LZ15; ZL15; ZS15; ZS16; WX17; ST17] and stochastic gradient descent [Che+14; Qia+16]. Similar optimization algorithms have been studied for reinforcement learning [WC16; Du+17]. Recently, a "doubly" stochastic method that randomizes both primal and dual updates was proposed for strongly convex bilinear saddle point problems [Yu+15]. For general saddle point problems, "doubly" stochastic gradient descent methods are discussed in [Nem+09; PB16], in which primal and dual variables are updated simultaneously based on the previous iterates and the current gradients.

Chapter 3: Convergence Analysis of ADMM

We now study the convergence of ADMM adaptive penalty parameters. We provide conditions on penalty parameters that guarantee convergence, and also a convergence rate. The issue of how to automatically tune penalty parameters effectively will be discussed in Chapter 4 and Chapter 5.

We prove for a slightly more general form of ADMM, where the penalty parameter is a diagonal matrix instead of a single number. To the best of our knowledge, this is the first worst-case convergence rate for ADMM with changing parameters. The theory results in this chapter has been published in [Xu+17c].

3.1 Generalized ADMM with diagonal penalty parameters

Let $T_k = \text{diag}(\tau_k^1, \ldots, \tau_k^p)$ be a diagonal matrix containing non-negative penalty parameters on iteration k. Define the norm $\|u\|_T^2 = u^T T u$. Using the notation defined above with $u = (u_1; \ldots; u_N) \in \mathbb{R}^{dN}$, we can write the ADMM steps as

$$u_{k+1} = \arg\min_u f(u) + \langle -Au, \lambda_k \rangle + 1/2 \|b - Au - Bv_k\|_{T_k}^2 \tag{3.1}$$

$$v_{k+1} = \arg\min_v g(v) + \langle -Bv, \lambda_k \rangle + 1/2 \|b - Au_{k+1} - Bv\|_{T_k}^2 \tag{3.2}$$

$$\lambda_{k+1} = \lambda_k + T_k(b - Au_{k+1} - Bv_{k+1}). \tag{3.3}$$

When using a diagonal penalty matrix, the generalized residuals become

$$\begin{cases} r_k = b - Au_k - Bu_k \\ d_k = A^T T_k B(v_k - v_{k-1}). \end{cases} \tag{3.4}$$

The sequel contains a convergence proof for generalized ADMM with adaptive penalty matrix T_k. Our proof is inspired by the variational inequality (VI) approach in [He+00; HY12b; HY15].

3.2 Preliminaries

Notation. We use the following notation to simplify the discussions. Define the combined variables $y = (u; v) \in \mathbb{R}^{n+m}$ and $z = (u; v; \lambda) \in \mathbb{R}^{n+m+p}$, and denote iterates as $y_k = (u_k; v_k)$ and $z_k = (u_k; v_k; \lambda_k)$. Let $y^\star$ and $z^\star$ denote optimal primal/dual solutions. Further define $\Delta z_k^+ = (\Delta u_k^+; \Delta v_k^+; \Delta \lambda_k^+) := z_{k+1} - z_k$ and $\Delta z_k^* = (\Delta u_k^*; \Delta v_k^*; \Delta \lambda_k^*) := z^\star - z_k$. Set

$$\phi(y) = f(u) + g(v), \;\; F(z) = \begin{pmatrix} -A^T\lambda \\ -B^T\lambda \\ Au + Bv - b \end{pmatrix},$$

$$H_k = \begin{pmatrix} 0 & 0 & 0 \\ 0 & B^T T_k B & 0 \\ 0 & 0 & (T_k)^{-1} \end{pmatrix}, \; M_k = \begin{pmatrix} I_n & 0 & 0 \\ 0 & I_m & 0 \\ 0 & -T_k B & I_p \end{pmatrix}.$$

Note that $F(z)$ is a monotone operator satisfying $\forall z, z', (z-z')^T(F(z)-F(z')) \geq 0$. We introduce intermediate variable $\tilde{z}_{k+1} = (u_{k+1}; v_{k+1}; \hat{\lambda}_{k+1})$, where $\hat{\lambda}_{k+1} = \lambda_k + T_k(b - Au_{k+1} - Bv_k)$. We thus have

$$\Delta z_k^+ = M_k(\tilde{z}_{k+1} - z_k). \tag{3.5}$$

Variational inequality formulation. The optimal solution $z^\star$ of problem (5.156) satisfies the variational inequality (VI),

$$\forall z,\ \phi(y) - \phi(y^\star) + (z - z^\star)^T F(z^\star) \geq 0. \tag{3.6}$$

From the optimality conditions for the sub-steps (3.1, 3.2), we see that y_{k+1} satisfies the variational inequalities

$$\forall u, f(u) - f(u_{k+1}) + (u - u_{k+1})^T(A^T T_k(Au_{k+1} + Bv_k - b) - A^T\lambda_k) \geq 0 \tag{3.7}$$

$$\forall v, g(v) - g(v_{k+1}) + (v - v_{k+1})^T(B^T T_k(Au_{k+1} + Bv_{k+1} - b) - B^T\lambda_k) \geq 0, \tag{3.8}$$

which can be combined as

$$\phi(y) - \phi(y_{k+1}) + (z - \tilde{z}_{k+1})^T \left(F(\tilde{z}_{k+1}) + H_k\Delta z_k^+\right) \geq 0. \tag{3.9}$$

Lemmas. We present several lemmas to facilitate the proof of our main convergence theory, which extend previous results regarding ADMM [HY12b; HY15] to ADMM with a diagonal penalty matrix. Lemma 3.2.1 shows the difference between

iterates decreases as the iterates approach the true solution, while Lemma 3.2.2 implies a contraction in the VI sense. Full proofs are provided in appendix; Eq. (6.9) and Eq. (6.14) are supported using equations (3.6, 3.8, 3.9) and standard techniques, while Eq. (3.12) is proven from Eq. (6.14). Lemma 3.2.2 is supported by the relationship in Eq. (3.5).

Lemma 3.2.1. *The optimal solution* $z^\star = (u^\star; v^\star; \lambda^\star)$ *and sequence* $z_k = (u_k; v_k; \lambda_k)$ *of generalized ADMM satisfy*

$$(B\Delta v_k^+)^T \Delta\lambda_k^+ \geq 0, \tag{3.10}$$

$$\Delta z_{k+1}^* H_k \Delta z_k^+ \geq 0, \tag{3.11}$$

$$\|\Delta z_k^+\|_{H_k}^2 \leq \|\Delta z_k^*\|_{H_k}^2 - \|\Delta z_{k+1}^*\|_{H_k}^2. \tag{3.12}$$

Lemma 3.2.2. *The sequence* $\tilde{z}_k = (u_k; v_k; \hat{\lambda}_k)$ *and* $z_k = (u_k; v_k; \lambda_k)^T$ *from generalized ADMM satisfy,* $\forall z$,

$$(\tilde{z}_{k+1} - z)^T H_k \Delta z_k^+ \geq \frac{1}{2}(\|z_{k+1} - z\|_{H_k}^2 - \|z_k - z\|_{H_k}^2). \tag{3.13}$$

3.3 Convergence criteria

We provide a convergence analysis of ADMM with an adaptive diagonal penalty matrix by showing (i) the norm of the residuals converges to zero; (ii) the method attains a worst-case ergodic $O(1/k)$ convergence rate in the VI sense. The key idea of the proof is to bound the adaptivity of T^k so that ADMM is stable enough to converge, which is presented as the following assumption.

Assumption 3.3.1. *The adaptivity of the diagonal penalty matrix* $T_k = diag(\tau_{1,k}, \ldots, \tau_{p,k})$ *is bounded by*

$$\begin{aligned} &\sum_{k=1}^{\infty} (\eta_k)^2 < \infty, \ \text{where } (\eta_k)^2 = \max_{i \in \{1,\ldots,p\}} \{(\eta_{i,k})^2\}, \\ &(\eta_{i,k})^2 = \max\{\tau_{i,k}/\tau_{i,k-1} - 1, \tau_{i,k-1}/\tau_{i,k} - 1\}. \end{aligned} \tag{3.14}$$

We can apply Assumption 3.3.1 to verify that

$$\frac{1}{1+\eta_k^2} \leq \frac{\tau_{i,k}}{\tau_{i,k-1}} \leq 1 + \eta_k^2. \tag{3.15}$$

which is needed to prove Lemma 3.3.1.

Lemma 3.3.1. *Suppose Assumption 3.3.1 holds. Then* $z = (u; v; \lambda)$ *and* $z' = (u'; v'; \lambda')$ *satisfy,* $\forall z, z'$

$$\|z - z'\|_{H_k}^2 \leq (1 + (\eta_k)^2)\|z - z'\|_{H_{k-1}}^2. \tag{3.16}$$

Now we are ready to prove the convergence of generalized ADMM with adaptive penalty under Assumption 3.3.1. We prove the following quantity, which is a norm of the residuals, converges to zero.

$$\begin{aligned} \|\Delta z_k^+\|_{H_k}^2 =& \|B\Delta v_k^+\|_{T_k}^2 + \|\Delta\lambda_k^+\|_{(T_k)^{-1}}^2 \\ =& \|(A^T T_k)^\dagger d_k\|_{T_k}^2 + \|r_k\|_{T_k}^2, \end{aligned} \tag{3.17}$$

where $A^\dagger$ denotes generalized inverse of a matrix A. Note that $\|\Delta z_k^+\|_{H_k}^2$ converges

to zero only if $\|r_k\|$ and $\|d_k\|$ converge to zero, provided A and T_k are bounded.

Theorem 3.3.1. *Suppose Assumption 3.3.1 holds. Then the iterates* $z_k = (u_k; v_k; \lambda_k)$ *of generalized ADMM satisfy*

$$\lim_{k\to\infty} \|\Delta z_k^+\|_{H_k}^2 = 0. \tag{3.18}$$

Proof. Let $z = z_k, z' = z^\star$ in Lemma 3.3.1 to achieve

$$\|\Delta z_k^*\|_{H_k}^2 \le (1 + (\eta_k)^2)\|\Delta z_k^*\|_{H_{k-1}}^2. \tag{3.19}$$

Combine (3.19) with Lemma 3.2.1 (3.12) to get

$$\|\Delta z_k^+\|_{H_k}^2 \le (1 + (\eta_k)^2)\|\Delta z_k^*\|_{H_{k-1}}^2 - \|\Delta z_{k+1}^*\|_{H_k}^2. \tag{3.20}$$

Accumulate (3.20) for $k = 1$ to l,

$$\sum_{k=1}^{l} \prod_{t=k+1}^{l} (1 + (\eta^t)^2)\|\Delta z_k^+\|_{H_k}^2 \le \prod_{t=1}^{l}(1 + (\eta^t)^2)\|\Delta z_1^*\|_{H^0}^2 - \|\Delta z_{l+1}^*\|_{H^l}^2. \tag{3.21}$$

Then we have

$$\sum_{k=1}^{l} \|\Delta z_k^+\|_{H_k}^2 \le \prod_{t=1}^{l}(1 + (\eta^t)^2)\|\Delta z_1^*\|_{H^0}^2. \tag{3.22}$$

When $l \to \infty$, Assumption 3.3.1 suggests $\prod_{t=1}^{\infty}(1 + (\eta^t)^2) < \infty$, which means $\sum_{k=1}^{\infty} \|\Delta z_k^+\|_{H_k}^2 < \infty$. Hence $\lim_{k\to\infty} \|\Delta z_k^+\|_{H_k}^2 = 0$. □

We further exploit Assumption 3.3.1 and Lemma 3.3.1 to prove Lemma 3.3.2,

and combine VI (3.9), Lemma 3.2.2, and Lemma 3.3.2 to prove the $O(1/k)$ convergence rate in Theorem 3.3.2.

Lemma 3.3.2. *Suppose Assumption 3.3.1 holds. Then $z = (u; v; \lambda) \in \mathbb{R}^{m+n+p}$ and the iterates $z_k = (u_k; v_k; \lambda_k)$ of generalized ADMM satisfy, $\forall z$*

$$\sum_{k=1}^{l}(\|z - z_k\|_{H_k}^2 - \|z - z_k\|_{H_{k-1}}^2) \leq C_\eta^\Sigma C_\eta^\Pi(\|z - z^\star\|_{H^0}^2 + \|\Delta z_1^*\|_{H^0}^2) < \infty, \quad (3.23)$$

where $C_\eta^\Sigma = \sum_{k=1}^{\infty}(\eta_k)^2$, $C_\eta^\Pi = \prod_{t=1}^{\infty}(1 + (\eta^t)^2)$.

Theorem 3.3.2. *Suppose Assumption 3.3.1 holds. Consider the sequence $\tilde{z}^k = (u^k; v^k; \hat{\lambda}^k)$ of generalized ADMM and define $\bar{z}^l = \frac{1}{l}\sum_{k=1}^{l}\tilde{z}^k$. Then sequence $\bar{z}^l$ satisfies the convergence bound*

$$\phi(y) - \phi(\bar{y}^l) + (z - \bar{z}^l)^T F(\bar{z}^l) \geq -\frac{1}{2\,l}(\|z - z^0\|_{H^0}^2 + C_\eta^\Sigma C_\eta^\Pi\|z - z^\star\|_{H^0}^2 + C_\eta^\Sigma C_\eta^\Pi\|\Delta z_1^*\|_{H^0}^2). \quad (3.24)$$

Proof. We can verify with simple algebra that

$$(z - z')^T F(z) = (z - z')^T F(z'). \quad (3.25)$$

Apply (3.25) with $z' = \tilde{z}_{k+1}$, and combine VI (3.9) and Lemma 3.2.2 to get

$$\phi(y) - \phi(y_{k+1}) + (z - \tilde{z}_{k+1})^T F(z) \tag{3.26}$$

$$= \phi(y) - \phi(y_{k+1}) + (z - \tilde{z}_{k+1})^T F(\tilde{z}_{k+1}) \tag{3.27}$$

$$\geq (\tilde{z}_{k+1} - z)^T H_k \Delta z_k^+ \tag{3.28}$$

$$\geq \frac{1}{2}(\|z_{k+1} - z\|_{H_k}^2 - \|z_k - z\|_{H_k}^2). \tag{3.29}$$

Summing for $k = 0$ to $l - 1$ gives us

$$\begin{aligned} &\sum_{k=1}^{l} \phi(y) - \phi(y_k) + (z - \tilde{z}_k)^T F(z) \\ \geq &\frac{1}{2} \sum_{k=1}^{l} (\|z - z_k\|_{H_{k-1}}^2 - \|z - z_{k-1}\|_{H_{k-1}}^2). \end{aligned} \tag{3.30}$$

Since $\phi(y)$ is convex, the left hand side of (3.30) satisfies,

$$\begin{aligned} LHS &= l\,\phi(y) - \sum_{k=1}^{l} \phi(y_k) + (l\,z - \sum_{k=1}^{l} \tilde{z}_k)^T F(z) \\ &\leq l\,\phi(y) - l\,\phi(\bar{y}^l) + (l\,z - l\,\bar{z}^l)^T F(z). \end{aligned} \tag{3.31}$$

Applying Lemma 3.3.2, we see the right hand side satisfies,

$$\begin{aligned} RHS =& \frac{1}{2}\sum_{k=1}^{l}(\|z-z_k\|_{H_k}^2 - \|z-z_{k-1}\|_{H_{k-1}}^2)+ \\ & \frac{1}{2}\sum_{k=1}^{l}(\|z-z_k\|_{H_{k-1}}^2 - \|z-z_k\|_{H_k}^2) \end{aligned} \tag{3.32}$$

$$\begin{aligned} \geq & \frac{1}{2}(\|z-z^l\|_{H^l}^2 - \|z-z^0\|_{H^0}^2)+ \\ & -\frac{1}{2}C_\eta^\Sigma C_\eta^\Pi(\|z-z^\star\|_{H^0}^2 + \|\Delta z_1^*\|_{H^0}^2) \end{aligned} \tag{3.33}$$

$$\begin{aligned} \geq & -\frac{1}{2}(\|z-z^0\|_{H^0}^2 + C_\eta^\Sigma C_\eta^\Pi\|z-z^\star\|_{H^0}^2+ \\ & C_\eta^\Sigma C_\eta^\Pi\|\Delta z_1^*\|_{H^0}^2). \end{aligned} \tag{3.34}$$

Combining inequalities (3.30), (3.31) and (3.34), and letting $z' = \bar{z}_k$ in (3.25) yields the $O(1/k)$ convergence rate in (3.24) □

3.4 Appendix: proof of lemmas

Proof of Lemma 3.2.1 (6.9)

Proof. By using the updated dual variable λ_{k+1} in (3.3), VI (3.8) can be rewritten as

$$\forall v,\ g(v) - g(v_{k+1}) - (Bv - Bv_{k+1})^T\lambda_{k+1} \geq 0. \tag{3.35}$$

Similarly, in the previous iteration,

$$\forall v,\ g(v) - g(v_k) - (Bv - Bv_k)^T\lambda_k \geq 0. \tag{3.36}$$

Let $v = v_k$ in (3.35) and $v = v_{k+1}$ in (3.36), and sum the two inequalities together. We conclude

$$(Bv_{k+1} - Bv_k)^T(\lambda_{k+1} - \lambda_k) \geq 0. \tag{3.37}$$

□

Proof of Lemma 3.2.1 (6.14)

Proof. VI (3.9) can be rewritten as

$$\phi(y) - \phi(y_{k+1}) + (z - z_{k+1})^T \left(F(z_{k+1}) + \Omega(\Delta z_k^+, T_k)\right) \geq 0, \tag{3.38}$$

where $\Omega(\Delta z_k^+, T_k) = (-A^T T_k B \Delta v_k^+; 0; (T_k)^{-1}\Delta\lambda_k^+)$.

Let $y = y^\star, z = z^\star$ in VI (3.38), and $y = y_{k+1}, z = z_{k+1}$ in VI (3.6), and sum the two equalities together to get

$$\begin{aligned}(\Delta z_{k+1}^*)^T \Omega(\Delta z_k^+, T_k) \geq \\ (\Delta z_{k+1}^*)^T (F(z^\star) - F(z_{k+1})).\end{aligned} \tag{3.39}$$

Since $F(z)$ is monotone, the right hand side is non-negative. Now, substitute $\Omega(\Delta z_k^+, T_k)$ into (3.39) to get

$$-(A\Delta u_{k+1}^*)^T T_k (B\Delta v_k^+) + (\Delta\lambda_{k+1}^*)^T (T_k)^{-1}\Delta\lambda_k^+ \geq 0. \tag{3.40}$$

If we use the feasibility constraint of optimal solution ($Au^\star + Bv^\star = b$) and the dual

update formula (3.3), we have

$$T_k A \Delta u^*_{k+1} = \Delta \lambda^+_k - T_k B \Delta v^*_{k+1}. \tag{3.41}$$

Substitute this into (4.13) yields

$$(B\Delta v^*_{k+1})^T T_k B \Delta v^+_k + (\Delta \lambda^*_{k+1})^T (T_k)^{-1} \Delta \lambda^+_k \geq (B\Delta v^+_k)^T \Delta \lambda^+_k \tag{3.42}$$

The proof (6.14) is concluded by applying (6.9) to (3.42). □

Proof of Lemma 3.2.1 (3.12)

Proof.

$$\begin{aligned}
\|\Delta z^*_k\|^2_{H_k} &= \|z^\star - z_k\|^2_{H_k} && (3.43)\\
&= \|z^\star - z_{k+1} + z_{k+1} - z_k\|^2_{H_k} && (3.44)\\
&= \|\Delta z^*_{k+1} + \Delta z^+_k\|^2_{H_k} && (3.45)\\
&= \|\Delta z^*_{k+1}\|^2_{H_k} + \|\Delta z^+_k\|^2_{H_k} + 2(\Delta z^*_{k+1})^T H_k \Delta z^+_k && (3.46)\\
&\geq \|\Delta z^*_{k+1}\|^2_{H_k} + \|\Delta z^+_k\|^2_{H_k}. && (3.47)
\end{aligned}$$

Eq. (6.14) is used for the inequality in (3.47), and Eq. (3.12) is derived by rearranging the order of $\|\Delta z^*_k\|^2_{H_k} \geq \|\Delta z^*_{k+1}\|^2_{H_k} + \|\Delta z^+_k\|^2_{H_k}$. □

Proof of Lemma 3.2.2

Proof. Applying the observation

$$\begin{aligned}(a-b)^T H(c-d) = \frac{1}{2}(\|a-d\|_H^2 - \|a-c\|_H^2) \\ + \frac{1}{2}(\|c-b\|_H^2 - \|c-d\|_H^2),\end{aligned} \tag{3.48}$$

we have

$$(\tilde{z}_{k+1} - z)^T H_k \Delta z_k^+ = (\tilde{z}_{k+1} - z) H_k (z_{k+1} - z_k) \tag{3.49}$$

$$\begin{aligned}=&\frac{1}{2}(\|\tilde{z}_{k+1} - z_k\|_{H_k}^2 - \|\tilde{z}_{k+1} - z_{k+1}\|_{H_k}^2) + \\ &\frac{1}{2}(\|z_{k+1} - z\|_{H_k}^2 - \|z_k - z\|_{H_k}^2).\end{aligned} \tag{3.50}$$

We now consider

$$\|\tilde{z}_{k+1} - z_{k+1}\|_{H_k}^2 = \|\tilde{z}_{k+1} - z_k + z_k - z_{k+1}\|_{H_k}^2 \tag{3.51}$$

$$=\|\tilde{z}_{k+1} - z_k\|_{H_k}^2 + \|\Delta z_k^+\|_{H_k}^2 - 2(\tilde{z}_{k+1} - z_k)^T H_k \Delta z_k^+, \tag{3.52}$$

and get

$$\|\tilde{z}_{k+1} - z_k\|_{H_k}^2 - \|\tilde{z}_{k+1} - z_{k+1}\|_{H_k}^2 \tag{3.53}$$

$$=2(\tilde{z}_{k+1} - z_k)^T H_k \Delta z_k^+ - \|\Delta z_k^+\|_{H_k}^2. \tag{3.54}$$

We then substitute Δz_k^+ with $M_k(\tilde{z}_{k+1} - z_k)$ in (3.5),

$$\|\tilde{z}_{k+1} - z_k\|_{H_k}^2 - \|\tilde{z}_{k+1} - z_{k+1}\|_{H_k}^2 \tag{3.55}$$

$$=(\tilde{z}_{k+1} - z_k)^T(2I - M_k)^T H_k M_k(\tilde{z}_{k+1} - z_k) \tag{3.56}$$

$$=\|\hat{\lambda}_{k+1} - \lambda_k\|_{(T_k)^{-1}}^2 \geq 0. \tag{3.57}$$

Combining (3.50) and (3.57), we conclude

$$(\tilde{z}_{k+1} - z)^T H_k \Delta z_k^+ \geq \frac{1}{2}(\|z_{k+1} - z\|_{H_k}^2 - \|z_k - z\|_{H_k}^2). \tag{3.58}$$

□

Proof of Lemma 3.3.1

Proof. Assumption 3.3.1 implies (3.15), which suggests the diagonal matrices T_k and T_{k-1} satisfy

$$\begin{aligned} T_k \leq&(1 + (\eta_k)^2)T_{k-1} \\ (T_k)^{-1} \leq&(1 + (\eta_k)^2)(T_{k-1})^{-1}. \end{aligned} \tag{3.59}$$

Then we have

$$\|z - z'\|^2_{H_k} \tag{3.60}$$

$$= \|B(v - v')\|^2_{T_k} + \|\lambda - \lambda'\|^2_{(T_k)^{-1}} \tag{3.61}$$

$$\leq (1 + (\eta_k)^2)(\|B(v - v')\|^2_{T_{k-1}} + \|\lambda - \lambda'\|^2_{(T_{k-1})^{-1}}) \tag{3.62}$$

$$\leq (1 + (\eta_k)^2)\|z - z'\|^2_{H_{k-1}}. \tag{3.63}$$

The inequality (3.59) is used to get from (3.61) to (3.62). □

Proof of Lemma 3.3.2

Proof. From (3.20) we know

$$\|\Delta z_k^+\|^2_{H_k} + \|\Delta z^*_{k+1}\|^2_{H_k} \leq (1 + (\eta_k)^2)\|\Delta z^*_k\|^2_{H_{k-1}}. \tag{3.64}$$

Hence

$$\|\Delta z^*_{k+1}\|^2_{H_k} \leq (1 + (\eta_k)^2)\|\Delta z^*_k\|^2_{H_{k-1}} \tag{3.65}$$

$$\leq \prod_{t=1}^{k} (1 + (\eta^t)^2)\|\Delta z^*_1\|^2_{H^0} \tag{3.66}$$

$$\leq \prod_{t=1}^{\infty} (1 + (\eta^t)^2)\|\Delta z^*_1\|^2_{H^0} \tag{3.67}$$

$$= C^{\Pi}_{\eta}\, \|\Delta z^*_1\|^2_{H^0} < \infty. \tag{3.68}$$

Let $z' = z^\star$ in Lemma 3.3.1, we have

$$\|z - z^\star\|_{H_k}^2 \leq (1 + (\eta_k)^2)\|z - z^\star\|_{H_{k-1}}^2 \tag{3.69}$$

$$\leq \prod_{t=1}^{k}(1 + (\eta^t)^2)\|z - z^\star\|_{H^0}^2 \tag{3.70}$$

$$\leq \prod_{t=1}^{\infty}(1 + (\eta^t)^2)\|z - z^\star\|_{H^0}^2 \tag{3.71}$$

$$= C_\eta^\Pi \|z - z^\star\|_{H^0}^2 < \infty. \tag{3.72}$$

Let $z' = z_k$ in Lemma 3.3.1, we have

$$\|z - z_k\|_{H_k}^2 \leq (1 + (\eta_k)^2)\|z - z_k\|_{H_{k-1}}^2. \tag{3.73}$$

Then we have

$$\sum_{k=1}^{l}(\|z-z_k\|_{H_k}^2-\|z-z_k\|_{H_{k-1}}^2) \tag{3.74}$$

$$\leq\sum_{k=1}^{l}(\eta_k)^2\|z-z_k\|_{H_{k-1}}^2 \tag{3.75}$$

$$=\sum_{k=1}^{l}(\eta_k)^2\|z-z^\star+z^\star-z_k\|_{H_{k-1}}^2 \tag{3.76}$$

$$\leq\sum_{k=1}^{l}(\eta_k)^2(\|z-z^\star\|_{H_{k-1}}^2+\|\Delta z_k^*\|_{H_{k-1}}^2) \tag{3.77}$$

$$\leq\sum_{k=1}^{l}(\eta_k)^2(C_\eta^\Pi\|z-z^\star\|_{H^0}^2+C_\eta^\Pi\|\Delta z_1^*\|_{H^0}^2) \tag{3.78}$$

$$\leq\sum_{k=1}^{\infty}(\eta_k)^2(C_\eta^\Pi\|z-z^\star\|_{H^0}^2+C_\eta^\Pi\|\Delta z_1^*\|_{H^0}^2) \tag{3.79}$$

$$=C_\eta^\Sigma(C_\eta^\Pi\|z-z^\star\|_{H^0}^2+C_\eta^\Pi\|\Delta z_1^*\|_{H^0}^2) \tag{3.80}$$

$$=C_\eta^\Sigma C_\eta^\Pi(\|z-z^\star\|_{H^0}^2+\|\Delta z_1^*\|_{H^0}^2)<\infty. \tag{3.81}$$

□

Chapter 4: Adaptive ADMM

The *alternating direction method of multipliers* (ADMM) is an invaluable element of the modern optimization toolbox. ADMM decomposes complex optimization problems into sequences of simpler subproblems, often solvable in closed form; its simplicity, flexibility, and broad applicability, make ADMM a state-of-the-art solver in machine learning, signal processing, and many other areas [Boy+11].

It is well known that the efficiency of ADMM hinges on the careful selection of a *penalty parameter*, which needs to be manually tuned by users for their particular problem instances. In contrast, for gradient descent and proximal-gradient methods, adaptive (*i.e.* automated) stepsize selection rules have been proposed, which essentially dispense with user oversight and dramatically boost performance [BB88; Fle05; Gol+14a; Wri+09c; Zho+06].

In this chapter, we propose to automate and speed up ADMM by using stepsize selection rules adapted from the gradient descent literature, namely the Barzilai-Borwein "spectral" method for smooth unconstrained problems [BB88; Fle05]. Since ADMM handles multi-term objectives and linear constraints, it is not immediately obvious how to adopt such rules. The keystone of our approach is to analyze the dual of the ADMM problem, which can be written without constraints. To ensure

reliability of the method, we develop a correlation criterion that safeguards it against inaccurate stepsize choices. The resulting *adaptive ADMM* (AADMM) algorithm is fully automated and fairly insensitive to the initial stepsize, as testified for by a comprehensive set of experiments. We propose AADMM in [Xu+17b], and provide more comprehensive studies in [Xu+17e].

4.1 Background and related work

4.1.1 Parameter tuning and adaptation

Relatively little work has been done on automating ADMM, *i.e.*, on adaptively choosing τ_k. In the particular case of a strictly convex quadratic objective, criteria for choosing an optimal constant penalty have been recently proposed by Ghadimi et al. [Gha+15] and Raghunathan and Di Cairano [RDC14]. Lin et al. [Lin+11] proposed a non-increasing sequence for the linearization parameter in "linearized" ADMM; however, they do not address the question of how to choose the penalty parameter in ADMM or its variants.

Residual balancing (RB) [He+00; Boy+11] is the only available adaptive method for general form problems (5.156); it is based on the following observation: increasing τ_k strengthens the penalty term, yielding smaller primal residuals but larger dual ones; conversely, decreasing τ_k leads to larger primal and smaller dual residuals. As both residuals must be small at convergence, it makes sense to "balance" them, *i.e.*, tune τ_k to keep both residuals of similar magnitude. A simple

scheme for this goal is

$$\tau_{k+1} = \begin{cases} \eta\tau_k & \text{if } \|r_k\|_2 > \mu\|d_k\|_2 \\ \tau_k/\eta & \text{if } \|d_k\|_2 > \mu\|r_k\|_2 \\ \tau_k & \text{otherwise,} \end{cases} \tag{4.1}$$

with $\mu > 1$ and $\eta > 1$ [Boy+11]. RB has recently been adapted to distributed optimization [Son+16] and other primal-dual splitting methods [Gol+15]. ADMM with adaptive penalty is not guaranteed to converge, unless τ_k is fixed after a finite number of iterations [He+00].

Despite some practical success of the RB idea, it suffers from several flaws. The relative size of the residuals depends on the scaling of the problem; e.g., with the change of variable $u \leftarrow 10u$, problem (5.156) can be re-scaled so that ADMM produces an equivalent sequence of iterates with residuals of very different magnitudes. Consequently, RB criteria are arbitrary in some cases, and their performance varies wildly with different problem scalings (see Section 4.3.3). Furthermore, the penalty parameter may adapt slowly if the initial value is far from optimal. Finally, without a careful choice of η and μ, the algorithm may fail to converge unless adaptivity is turned off [He+00].

4.1.2 Dual interpretation of ADMM

We now explain the close relationship between ADMM and *Douglas-Rachdord splitting* (DRS) [EB92; Ess09; Gol+14b], which plays a central role in the proposed

approach. The starting observation is that the dual of problem (5.156) has the form

$$\min_{\zeta \in \mathbb{R}^p} \underbrace{H^*(A^T\zeta) - \langle \zeta, b\rangle}_{\hat{H}(\zeta)} + \underbrace{G^*(B^T\zeta)}_{\hat{G}(\zeta)}, \tag{4.2}$$

where F^* denotes the Fenchel conjugate of F, defined as $F^*(y) = \sup_x \langle x, y\rangle - F(x)$ [Roc70].

The DRS algorithm solves (4.2) by generating two sequences $(\zeta_k)_{k\in\mathbb{N}}$ and $(\hat{\zeta}_k)_{k\in\mathbb{N}}$ according to

$$0 \in \frac{\hat{\zeta}_{k+1} - \zeta_k}{\tau_k} + \partial\hat{H}(\hat{\zeta}_{k+1}) + \partial\hat{G}(\zeta_k) \tag{4.3}$$

$$0 \in \frac{\zeta_{k+1} - \zeta_k}{\tau_k} + \partial\hat{H}(\hat{\zeta}_{k+1}) + \partial\hat{G}(\zeta_{k+1}), \tag{4.4}$$

where we use the standard notation $\partial F(x)$ for the subdifferential of F evaluated at x [Roc70].

Referring back to ADMM in (5.157)–(5.159), and defining $\hat{\lambda}_{k+1} = \lambda_k + \tau_k(b - Au_{k+1} - Bv_k)$, the optimality condition for the minimization in (5.157) is

$$0 \in \partial H(u_{k+1}) - A^T \underbrace{(\lambda_k + \tau_k(b - Au_{k+1} - Bv_k))}_{\hat{\lambda}_{k+1}}$$

which is equivalent to $A^T\hat{\lambda}_{k+1} \in \partial H(u_{k+1})$, thus[1] $u_{k+1} \in \partial H^*(A^T\hat{\lambda}_{k+1})$. A similar argument using the optimality condition for (5.158) leads to $v_{k+1} \in \partial G^*(B^T\lambda_{k+1})$.

[1] An important property relating F and F^* is that $y \in \partial H(x)$ if and only if $x \in \partial H^*(y)$ [Roc70].

Recalling (4.2), we arrive at

$$Au_{k+1} - b \in \partial \hat{H}(\hat{\lambda}_{k+1}) \;\text{ and }\; Bv_{k+1} \in \partial \hat{G}(\lambda_{k+1}). \tag{4.5}$$

Using these identities, we finally have

$$\begin{aligned} \hat{\lambda}_{k+1} &= \lambda_k + \tau_k(b - Au_{k+1} - Bv_k) \\ &\in \lambda_k - \tau_k\big(\partial \hat{H}(\hat{\lambda}_{k+1}) + \partial \hat{G}(\lambda_k)\big) \qquad (4.6) \\ \lambda_{k+1} &= \lambda_k + \tau_k(b - Au_{k+1} - Bv_{k+1}) \\ &\in \lambda_k - \tau_k\big(\partial \hat{H}(\hat{\lambda}_{k+1}) + \partial \hat{G}(\lambda_{k+1})\big), \qquad (4.7) \end{aligned}$$

showing that the sequences $(\lambda_k)_{k\in\mathbb{N}}$ and $(\hat{\lambda}_k)_{k\in\mathbb{N}}$ satisfy the same conditions (4.3) and (4.4) as $(\zeta_k)_{k\in\mathbb{N}}$ and $(\hat{\zeta}_k)_{k\in\mathbb{N}}$, thus proving that ADMM for problem (5.156) is equivalent to DRS for its dual (4.2).

4.1.3 Spectral stepsize selection

The classical gradient descent step for unconstrained minimization of a smooth function $F\colon \mathbb{R}^n \to \mathbb{R}$ has the form $x_{k+1} = x_k - \tau_k \nabla F(x_k)$. Spectral gradient methods, pioneered by Barzilai and Borwein (BB) [BB88], adaptively choose the stepsize τ_k to achieve fast convergence.

In a nutshell, the standard (there are variants) BB method sets $\tau_k = 1/\alpha_k$, with α_k chosen such that $\alpha_k I$ mimics the Hessian of F over the last step, seeking a

quasi-Newton step. A least squares criterion yields

$$\alpha_k = \underset{\alpha \in \mathbb{R}}{\operatorname{argmin}} \|\nabla F(x_k) - \nabla F(x_{k-1}) - \alpha(x_k - x_{k-1})\|_2^2, \tag{4.8}$$

which is an estimate of the curvature of F across the previous step of the algorithm. BB gradient methods often dramatically outperform those with constant stepsize [Fle05; Zho+06] and have been generalized to handle non-differentiable problems via proximal gradient methods [Wri+09c; Gol+14a]. Finally, notice that (4.8) is equivalent to approximating the gradient $\nabla F(x_k)$ as a linear function of x_k,

$$\nabla F(x_k) \approx \nabla F(x_{k-1}) + \alpha_k(x_k - x_{k-1}) = \alpha_k\, x_k + a_k, \tag{4.9}$$

where $a_k = \nabla F(x_{k-1}) - \alpha_k\, x_{k-1}$. The observation that a local linear approximation of the gradient has an optimal parameter equal to the inverse of the BB stepsize will play an important role below.

4.2 Spectral penalty parameters

Inspired by the BB method, we propose a spectral penalty parameter selection method for ADMM. We first derive a spectral stepsize rule for DRS, and then adapt this rule to ADMM. Finally, we discuss safeguarding rules to prevent unexpected behavior when curvature estimates are inaccurate.

4.2.1 Spectral stepsize for DRS

Consider the dual problem (4.2). Following the observation in (4.9) about the BB method, we approximate $\partial\hat{H}$ and $\partial\hat{G}$ at iteration k as linear functions,

$$\partial\hat{H}(\hat{\zeta}) = \alpha_k\,\hat{\zeta} + \Psi_k \qquad \text{and} \qquad \partial\hat{G}(\zeta) = \beta_k\,\zeta + \Phi_k, \tag{4.10}$$

where $\alpha_k > 0$, $\beta_k > 0$ are local curvature estimates of dual functions $\hat{H}$ and $\hat{G}$, respectively, and $\Psi_k, \Phi_k \subset \mathbb{R}^p$. Once we obtain these curvature estimates, we will be able to exploit the following proposition.

Proposition 4.2.1 (Spectral DRS)**.** *Suppose the DRS steps (4.3)–(4.4) are applied to problem (4.2), where (omitting the subscript k from $\alpha_k, \beta_k, \Psi_k, \Phi_k$ to lighten the notation in what follows)*

$$\partial\hat{H}(\hat{\zeta}) = \alpha\,\hat{\zeta} + \Psi \qquad \textit{and} \qquad \partial\hat{G}(\zeta) = \beta\,\zeta + \Phi.$$

Then, the minimal residual of $\hat{H}(\zeta_{k+1}) + \hat{G}(\zeta_{k+1})$ is obtained by setting $\tau_k = 1/\sqrt{\alpha\,\beta}$.

Proof. Inserting (4.10) into the DRS step (4.3)–(4.4) yields

$$0 \in \frac{\hat{\zeta}_{k+1} - \zeta_k}{\tau} + (\alpha\,\hat{\zeta}_{k+1} + \Psi) + (\beta\,\zeta_k + \Phi), \tag{4.11}$$

$$0 \in \frac{\zeta_{k+1} - \zeta_k}{\tau} + (\alpha\,\hat{\zeta}_{k+1} + \Psi) + (\beta\,\zeta_{k+1} + \Phi). \tag{4.12}$$

From (4.11)–(4.12), we can explicitly get the update for $\hat{\zeta}_{k+1}$ as

$$\hat{\zeta}_{k+1} = \frac{1-\beta\,\tau}{1+\alpha\,\tau}\zeta_k - \frac{a\tau+b\tau}{1+\alpha\,\tau}, \tag{4.13}$$

where $a \in \Psi$ and $b \in \Phi$, and for ζ_{k+1} as

$$\zeta_{k+1} = \frac{1}{1+\beta\,\tau}\zeta_k - \frac{\alpha\,\tau}{1+\beta\,\tau}\hat{\zeta}_{k+1} - \frac{a\,\tau+b\tau}{1+\beta\,\tau} \tag{4.14}$$

$$= \frac{(1+\alpha\,\beta\,\tau^2)\zeta_k - (a+b)\tau}{(1+\alpha\,\tau)(1+\beta\,\tau)}, \tag{4.15}$$

where the second equality results from using the expression for $\hat{\zeta}_{k+1}$ in (4.13).

The residual r_{DR} at ζ_{k+1} is simply the magnitude of the subgradient (corresponding to elements $a \in \Psi$ and $b \in \Phi$) of the objective that is given by

$$r_{DR} = \|(\alpha+\beta)\zeta_{k+1} + (a+b)\|_2 \tag{4.16}$$

$$= \frac{1+\alpha\,\beta\,\tau^2}{(1+\alpha\,\tau)(1+\beta\,\tau)}\|(\alpha+\beta)\zeta_k + (a+b)\|_2, \tag{4.17}$$

where ζ_{k+1} in (4.17) was substituted with (4.15). The optimal stepsize τ_k minimizes the residual

$$\tau_k = \arg\min_{\tau} r_{\mathrm{DR}} = \arg\max_{\tau} \frac{(1+\alpha\,\tau)(1+\beta\,\tau)}{1+\alpha\,\beta\,\tau^2} \tag{4.18}$$

$$= \arg\max_{\tau} \frac{(\alpha+\beta)\tau}{1+\alpha\beta\tau^2} \tag{4.19}$$

$$= 1/\sqrt{\alpha\beta}. \tag{4.20}$$

Finally (recovering the iteration subscript k), notice that $\tau_k = (\hat{\alpha}_k \hat{\beta}_k)^{1/2}$, where $\hat{\alpha}_k = 1/\alpha_k$ and $\hat{\beta}_k = 1/\beta_k$ are the spectral gradient descent stepsizes for $\hat{H}$ and $\hat{G}$, at $\hat{\zeta}_k$ and ζ_k, respectively. □

Proposition 4.2.1 shows how to adaptively choose τ_k: begin by obtaining linear estimates of the subgradients of the two terms in the dual objective (4.2); the geometric mean of these optimal gradient descent stepsizes is then the optimal DRS stepsize, thus also the optimal ADMM penalty parameter, due to the equivalence shown in Subsection 4.1.2.

4.2.2 Spectral stepsize estimation

We now address the estimation of $\hat{\alpha}_k = 1/\alpha_k$ and $\hat{\beta}_k = 1/\beta_k$. These curvature parameters are estimated based on the results from iteration k and an older iteration $k_0 < k$. Noting (4.5), we define

$$\Delta\hat{\lambda}_k := \hat{\lambda}_k - \hat{\lambda}_{k_0}$$
$$\Delta\hat{H}_k := \partial\hat{H}(\hat{\lambda}_k) - \partial\hat{H}(\hat{\lambda}_{k_0}) = A(u_k - u_{k_0}).$$

Assuming, as above, a linear model for $\partial\hat{H}$, we expect $\Delta\hat{H}_k \approx \alpha\,\Delta\hat{\lambda}_k + a$. As is typical in BB-type methods [BB88; Zho+06], α is estimated via one of the two least squares problems

$$\min_{\alpha} \|\Delta\hat{H}_k - \alpha\Delta\hat{\lambda}_k\|_2^2 \text{ or } \min_{\alpha} \|\alpha^{-1}\Delta\hat{H}_k - \Delta\hat{\lambda}_k\|_2^2.$$

The closed form solutions for the corresponding spectral stepsizes $\hat{\alpha}_k = 1/\alpha_k$ are, respectively,

$$\hat{\alpha}_k^{\text{SD}} = \frac{\langle \Delta\hat{\lambda}_k, \Delta\hat{\lambda}_k \rangle}{\langle \Delta\hat{H}_k, \Delta\hat{\lambda}_k \rangle} \text{ and } \hat{\alpha}_k^{\text{MG}} = \frac{\langle \Delta\hat{H}_k, \Delta\hat{\lambda}_k \rangle}{\langle \Delta\hat{H}_k, \Delta\hat{H}_k \rangle}, \tag{4.21}$$

where, following Zhou et al. [Zho+06], SD stands for *steepest descent* and MG for *minimum gradient*. The Cauchy-Schwarz inequality implies that $\hat{\alpha}_k^{\text{SD}} \geq \hat{\alpha}_k^{\text{MG}}$. Rather than choosing one or the other, we suggest the hybrid stepsize rule proposed by Zhou et al. [Zho+06],

$$\hat{\alpha}_k = \begin{cases} \hat{\alpha}_k^{\text{MG}} & \text{if } \; 2\,\hat{\alpha}_k^{\text{MG}} > \hat{\alpha}_k^{\text{SD}} \\ \hat{\alpha}_k^{\text{SD}} - \hat{\alpha}_k^{\text{MG}}/2 & \text{otherwise.} \end{cases} \tag{4.22}$$

The spectral stepsize $\hat{\beta}_k = 1/\beta_k$ is similarly set to

$$\hat{\beta}_k = \begin{cases} \hat{\beta}_k^{\text{MG}} & \text{if } \; 2\,\hat{\beta}_k^{\text{MG}} > \hat{\beta}_k^{\text{SD}} \\ \hat{\beta}_k^{\text{SD}} - \hat{\beta}_k^{\text{MG}}/2 & \text{otherwise,} \end{cases} \tag{4.23}$$

where $\hat{\beta}_k^{\text{SD}} = \langle \Delta\lambda_k, \Delta\lambda_k \rangle / \langle \Delta\hat{G}_k, \Delta\lambda_k \rangle$, $\hat{\beta}_k^{\text{MG}} = \langle \Delta\hat{G}_k, \Delta\lambda_k \rangle / \langle \Delta\hat{G}_k, \Delta\hat{G}_k \rangle$, $\Delta\hat{G}_k = B(v_k - v_{k_0})$, and $\Delta\lambda_k = \lambda_k - \lambda_{k_0}$. It is important to note that $\hat{\alpha}_k$ and $\hat{\beta}_k$ are obtained from the iterates of ADMM, *i.e.*, the user is not required to supply the dual problem.

4.2.3 Safeguarding

On some iterations, the linear models (for one or both subgradients) underlying the spectral stepsize choice may be very inaccurate. When this occurs, the least squares procedure may produce ineffective stepsizes. The classical BB method for unconstrained problems uses a line search to safeguard against unstable stepsizes resulting from unreliable curvature estimates. In ADMM, however, there is no notion of "stable" stepsize (any constant stepsizes is stable), thus line search methods are not applicable. Rather, we propose to safeguard the method by assessing the quality of the curvature estimates, and only updating the stepsize if the curvature estimates satisfy a reliability criterion.

Algorithm 1 Adaptive ADMM (AADMM)

Input: initialize v_0, λ_0, τ_0, $k_0 = 0$
  while not converge by residual check (5.161) **and** $k <$ maxiter **do**
    $u_{k+1} = \arg\min_u H(u) + \frac{\tau_k}{2}\|b - Au - Bv_k + \frac{\lambda_k}{\tau_k}\|_2^2$
    $v_{k+1} = \arg\min_v G(v) + \frac{\tau_k}{2}\|b - Au_{k+1} - Bv + \frac{\lambda_k}{\tau_k}\|_2^2$
    $\lambda_{k+1} \leftarrow \lambda_k + \tau_k(b - Au_{k+1} - Bv_{k+1})$
    if $\mathrm{mod}(k, T_f) = 1$ **then**
      $\hat{\lambda}_{k+1} = \lambda_k + \tau_k(b - Au_{k+1} - Bv_k)$
      Estimate spectral stepsizes $\hat{\alpha}_{k+1}, \hat{\beta}_{k+1}$ in (4.22, 4.23)
      Estimate correlations $\alpha_{k+1}^{\mathrm{cor}}, \beta_{k+1}^{\mathrm{cor}}$ in (4.24)
      Update τ_{k+1} in (5.151)
      $k_0 \leftarrow k$
    else
      $\tau_{k+1} \leftarrow \tau_k$
    end if
    $k \leftarrow k + 1$
  end while

The linear model (4.10) assumes the change in dual (sub)gradient is linearly proportional to the change in the dual variables. To test the validity of this assump-

tion, we measure the correlation between these quantities (equivalently, the cosine of their angle):

$$\alpha_k^{\text{cor}} = \frac{\langle \Delta \hat{H}_k, \Delta \hat{\lambda}_k \rangle}{\|\Delta \hat{H}_k\| \, \|\Delta \hat{\lambda}_k\|} \text{ and } \beta_k^{\text{cor}} = \frac{\langle \Delta \hat{G}_k, \Delta \lambda_k \rangle}{\|\Delta \hat{G}_k\| \, \|\Delta \lambda_k\|}. \tag{4.24}$$

The spectral stepsizes are updated only if the correlations indicate the estimation is credible enough. The safeguarded spectral adaptive penalty rule is

$$\tau_k = \begin{cases} \sqrt{\hat{\alpha}_k \hat{\beta}_k} & \text{if } \alpha_k^{\text{cor}} > \epsilon^{\text{cor}} \text{ and } \beta_k^{\text{cor}} > \epsilon^{\text{cor}} \\ \hat{\alpha}_k & \text{if } \alpha_k^{\text{cor}} > \epsilon^{\text{cor}} \text{ and } \beta_k^{\text{cor}} \leq \epsilon^{\text{cor}} \\ \hat{\beta}_k & \text{if } \alpha_k^{\text{cor}} \leq \epsilon^{\text{cor}} \text{ and } \beta_k^{\text{cor}} > \epsilon^{\text{cor}} \\ \tau_{k-1} & \text{otherwise,} \end{cases} \tag{4.25}$$

where ϵ^{cor} is a quality threshold for the curvature estimates, while $\hat{\alpha}_k$ and $\hat{\beta}_k$ are the stepsizes given by (4.22)–(4.23). Notice that (5.151) falls back to constant τ_k when both curvature estimates are deemed inaccurate.

4.2.4 Adaptive ADMM

Algorithm 1 shows the complete *adaptive ADMM* (AADMM). We suggest only updating the stepsize every T_f iterations. Safeguarding threshold $\epsilon^{\text{cor}} = 0.2$ and $T_f = 2$ generally perform well. The overhead of AADMM over ADMM is modest: only a few inner products plus the storage to keep one previous iterate.

4.2.5 Convergence

He et al. [He+00] proved that convergence is guaranteed for ADMM with adaptive penalty when either of the two following conditions are satisfied:

Assumption 4.2.1 (Bounded increasing)**.**

$$\sum_{k=1}^{\infty} \eta_k^2 < \infty, \ \textit{where } \eta_k = \sqrt{\max\{\frac{\tau_k}{\tau_{k-1}}, 1\} - 1}. \tag{4.26}$$

Assumption 4.2.2 (Bounded decreasing)**.**

$$\sum_{k=1}^{\infty} \theta_k^2 < \infty, \ \textit{where } \theta_k = \sqrt{\max\{\frac{\tau_{k-1}}{\tau_k}, 1\} - 1}. \tag{4.27}$$

Assumption 4.2.1 (Assumption 4.2.2) suggests that increasing (decreasing) of adaptive penalty is bounded. In the previous section, we have proved $O(1/k)$ convergence rate when both Assumption 4.2.1 and Assumption 4.2.2 are satisfied. In practice, these conditions can be satisfied by turning off adaptivity after a finite number of steps, which we have found unnecessary in our experiments with AADMM.

4.3 Experiments

4.3.1 Experimental setting

We consider several applications to demonstrate the effectiveness of the proposed AADMM. We focus on statistical problems involving non-differentiable objectives: linear regression with elastic net regularization [Efr+04; Gol+14b], low

Application	Dataset	Vanilla ADMM	Fast ADMM	Residual balance	**Adaptive ADMM**
Elastic net regression	Synthetic	2000+ (1.64)	263 (.270)	111 (.129)	**43 (.046)**
	Boston	2000+ (2.19)	208 (.106)	54 (.023)	**17 (.011)**
	Diabetes	594 (.269)	947 (.848)	28 (.020)	**10 (.005)**
	Leukemia	2000+ (22.9)	2000+ (24.2)	1737 (19.3)	**152 (1.70)**
	Prostate	548 (.293)	139 (.049)	29 (.015)	**16 (.012)**
	Servo	142 (.040)	44 (.017)	27 (.012)	**13 (.007)**
Low rank least squares	Synthetic	543(31.3)	129(7.30)	75(5.59)	**13(.775)**
	Madelon	1943(925)	193(89.6)	133(60.9)	**27(12.8)**
	Sonar	1933(9.12)	313(1.51)	102(.466)	**31(.160)**
	Splice	1704(38.2)	189(4.25)	92(2.04)	**18(.413)**
QP and dual SVM	Synthetic	439 (6.15)	535 (7.8380)	232 (3.27)	**71 (.984)**
	Madelon	100 (14.0)	57 (8.14)	28 (4.12)	**19 (2.64)**
	Sonar	139 (.227)	43 (.075)	37 (.069)	**28 (.050)**
	Splice	149 (4.9)	47 (1.44)	39 (1.27)	**20 (.681)**
Basis pursuit	Synthetic	163 (.027)	2000+ (.310)	159 (.031)	**114 (.026)**
	Human1	2000+ (2.35)	2000+ (2.41)	839 (.990)	**503 (.626)**
	Human2	2000+ (2.26)	2000+ (2.42)	875 (1.03)	**448 (.554)**
	Human3	2000+ (2.29)	2000+ (2.44)	713 (.855)	**523 (.641)**
Consensus logistic regression	Synthetic	301 (3.36)	444 (3.54)	43 (.583)	**22 (.282)**
	Madelon	2000+ (205)	2000+ (166)	115 (42.1)	**23 (20.8)**
	Sonar	2000+ (33.5)	2000+ (47)	106 (2.82)	**90 (1.64)**
	Splice	2000+ (29.1)	2000+ (43.7)	86 (1.91)	**22 (.638)**
	News20	69 (5.91e3)	32 (3.45e3)	18 (1.52e3)	**16 (1.2e3)**
	Rcv1	38 (177)	23 (122)	**13 (53.0)**	**12 (53.9)**
	Realsim	1000+ (2.73e3)	1000+ (1.86e3)	121 (558)	**22 (118)**
Semidefinite programming	hamming-7-5-6	455(1.78)	2000+(8.60)	1093(4.21)	**284(1.11)**
	hamming-8-3-4	418(6.38)	2000+(29.1)	1071(16.5)	**118(2.02)**
	hamming-9-5-6	2000+(187)	2000+(187)	1444(131)	**481(53.1)**
	hamming-9-8	2000+(162)	2000+(159)	1247(97.2)	**594(52.7)**
	hamming-10-2	2000+(936)	2000+(924)	1194(556)	**391(193)**
	hamming-11-2	2000+(6.43e3)	2000+(6.30e3)	1203(4.15e3)	**447(1.49e3)**

Table 4.1: Iterations (and runtime in seconds) for the various algorithms and applications described in the text. Absence of convergence after n iterations is indicated as $n+$. AADMM is the proposed Algorithm 1.

rank least squares [YY13; Xu+15], quadratic programming (QP) [Boy+11; Gha+15; Gol+14b; RDC14], basis pursuit [Boy+11; Gol+14b], consensus ℓ_1-regularized logistic regression [Boy+11], and semidefinite programming [BM03; Wen+10]. We use both synthetic and benchmark datasets (obtained from the UCI repository and the LIBSVM page) used by Efron et al. [Efr+04], Lee et al. [Lee+06], Liu et al. [Liu+09], Schmidt et al. [Sch+07], and Wright et al. [Wri+09c], and Zou and Hastie [ZH05]. For the small and medium sized datasets, the features are standardized to zero mean and unit variance, whereas for the large and sparse datasets the features are scaled to be in $[-1, 1]$.

For comparison, we implemented *vanilla* ADMM (fixed stepsize), fast ADMM with a restart strategy [Gol+14b], and ADMM with residual balancing [Boy+11; He+00], using (4.1) with $\mu = 10$ and $\eta = 2$, and adaptivity was turned off after 1000 iterations to guarantee convergence. The proposed AADMM is implemented as shown in Algorithm 1, with fixed parameters $\epsilon^{\text{cor}} = 0.2$ and $T_f = 2$.

We set the stopping tolerance to $\epsilon^{tol} = 10^{-5}, 10^{-3}$, and 0.05 for small, medium, and large scale problems, respectively. The initial penalty $\tau_0 = 0.1$ is used for all problems, except the canonical QP, where τ_0 is set to the value proposed for quadratic problems by Raghunathan and Di Cairano [RDC14]. For each problem, the same randomly generated initial variables v_0, λ_0 are used for ADMM and all the variants thereof.

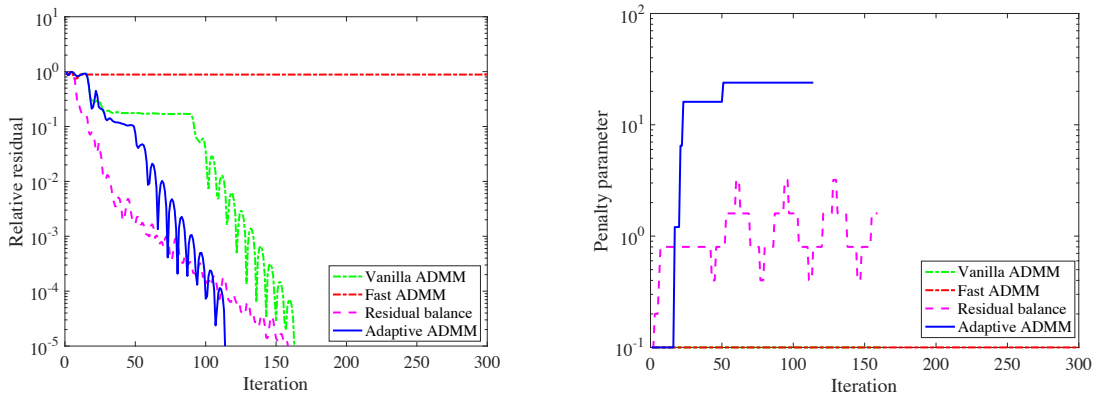


Figure 4.1: Relative residual (top) and penalty parameter (bottom) for the synthetic basis pursuit (BP) problem.

4.3.2 Convergence results

Table 4.1 reports the convergence speed of ADMM and its variants for the applications described in Section 2.4. Vanilla ADMM with fixed stepsize does poorly in practice: in 13 out of 23 realistic datasets, it fails to converge in the maximum number of iterations. Fast ADMM [Gol+14b] often outperforms vanilla ADMM, but does not compete with the proposed AADMM, which also outperforms residual balancing in all test cases except in the Rcv1 problem for consensus logistic regression.

Fig. 4.1 presents the relative residual (top) and penalty parameter (bottom) for the synthetic BP problem. The relative residual is defined as

$$\max\left\{\frac{\|r_k\|_2}{\max\{\|Au_k\|_2, \|Bv_k\|_2, \|b\|_2\}}, \frac{\|d_k\|_2}{\|A^T\lambda_k\|_2}\right\},$$

which is based on stopping criterion (5.161). Fast ADMM often restarts and is slow to converge. The penalty parameter chosen by RB oscillates. AADMM quickly

adapts the penalty parameter and converges fastest.

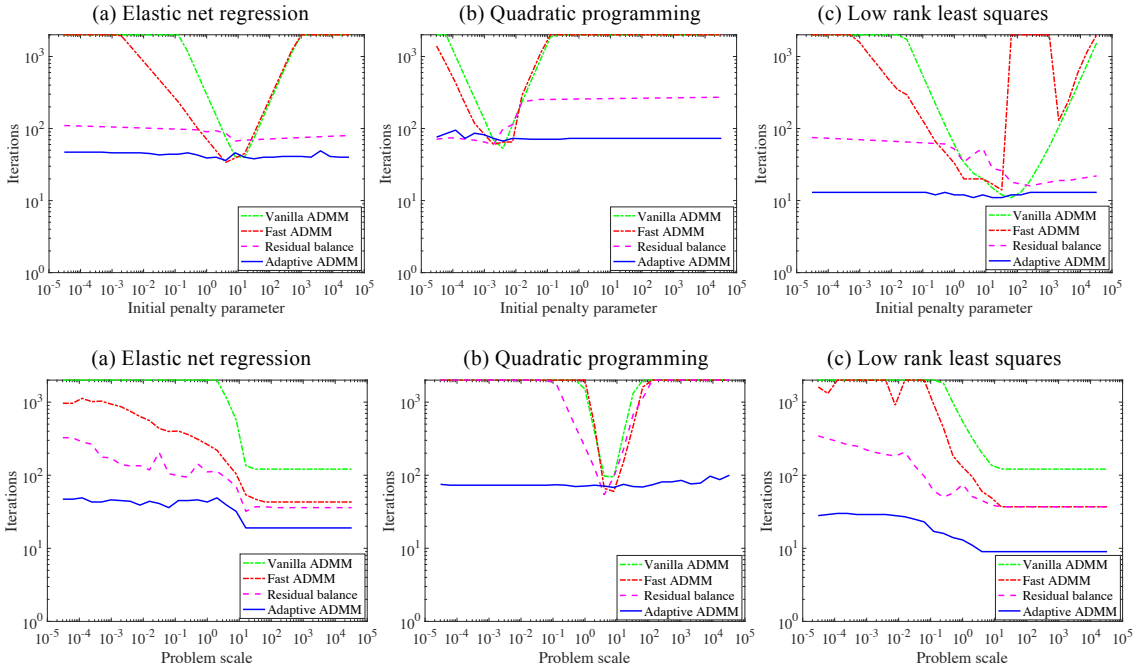


Figure 4.2: Top row: sensitivity of convergence speed to initial penalty parameter τ_0 for EN, QP, and LRLS. Bottom row: sensitivity to problem scaling s for EN, QP, and LRLS.

4.3.3 Sensitivity

We study the sensitivity of the different ADMM variants to problem scaling and initial penalty parameter (τ_0). Scaling sensitivity experiments were done by multiplying the measurement vector c by a scalar s. Fig. 5.10 presents iteration counts for a wide range of values of initial penalty τ_0 (top) and problems scale s (bottom) for EN regression, canonical QP, and LRLS with synthetic datasets. Fast ADMM and vanilla ADMM use the fixed initial penalty parameter τ_0, and are highly sensitive to this choice, as shown in Fig. 5.10; in contrast, AADMM is very stable with respect to τ_0 and the scale s.

Finally, Fig. 4.3 presents iteration counts when applying AADMM with various

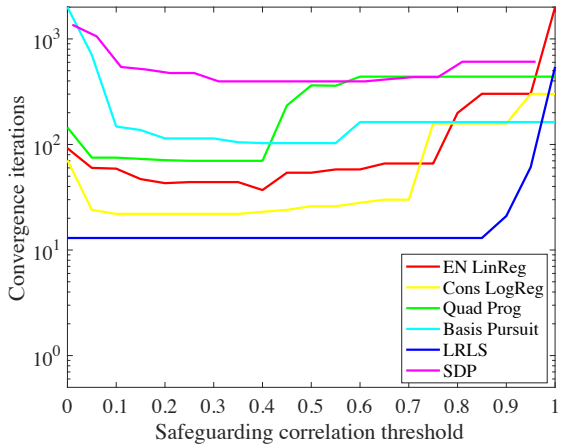


Figure 4.3: Sensitivity of convergence speed to safeguarding threshold ϵ^{cor} for proposed AADMM. Synthetic problems of various applications are studied. Best viewed in color.

safeguarding correlation thresholds ϵ^{cor}. When $\epsilon^{\text{cor}} = 0$ the new penalty value is always accepted, and when $\epsilon^{\text{cor}} = 1$ the penalty parameter is never changed. The proposed AADMM method is insensitive to ϵ^{cor} and performs well for a wide range of $\epsilon^{\text{cor}} \in [0.1, 0.4]$ for various applications.

4.4 Summarization

We have proposed *adaptive ADMM* (AADMM), a new variant of the popular ADMM algorithm that tackles one of its fundamental drawbacks: critical dependence on a penalty parameter that needs careful tuning. This drawback has made ADMM difficult to use by non-experts, thus AADMM has the potential to contribute to wider and easier applicability of this highly flexible and efficient optimization tool. Our approach imports and adapts the Barzilai-Borwein "spectral" stepsize method from the smooth optimization literature, tailoring it to the more general class of problems handled by ADMM. The cornerstone of our approach is

the fact that ADMM is equivalent to Douglas-Rachford splitting (DRS) applied to the dual problem, for which we develop a spectral stepsize selection rule; this rule is then translated into a criterion to select the penalty parameter of ADMM. A safeguarding function that avoids unreliable stepsize choices finally yields AADMM. Experiments on a comprehensive range of problems and datasets have shown that AADMM outperforms other variants of ADMM and is robust with respect to initial parameter choice and problem scaling.

Chapter 5: Variants of ADMM

In this chapter, we further exploit the adaptive schema in Chapter 4, and propose several variants of AADMM, including adaptive multi-block ADMM (AMADMM) [Xu+17e], adaptive relaxed ADMM (ARADMM) [Xu+17d], and adaptive consensus ADMM (ACADMM) [Xu+17c]. ARADMM achieves faster practical convergence speed by further automating the relaxed parameter for the extra interpolation step in ADMM. AMADMM works for problems with more than two primal splitting variables. ACADMM is specifically designed for formulated consensus problem for distributed computing. We study the theoretical convergence of ARADMM and ACADMM, while the theoretical guarantee of AMADMM remains an open problem. At the end of this chapter (Section 5.4), we present empirical results on applying AADMM to several nonconvex problems [Xu+16a]. Our experiments suggest the penalty parameter plays an even more important role for nonconvex problems, and our AADMM not only converges fast but also arrives at good stationary point.

5.1 Adaptive Multi-block ADMM

5.1.1 Residual balancing for multi-block ADMM

Based on the stop criterion in (2.17), a residual balancing criterion for multi-block ADMM is proposed as

$$\tau_{k+1} = \begin{cases} \eta\tau_k & \text{if} \quad \max_i\{\|d_{i,k}\|_2\} > \mu\|r_k\|_2 \\ \tau_k/\eta & \text{if} \quad \|r_k\|_2 > \mu \max_i\{\|d_{i,k}\|_2\} \\ \tau_k & \text{otherwise,} \end{cases} \tag{5.1}$$

for parameters $\mu > 1$ and $\eta > 1$. Defining normalized residuals as

$$r_k^{norm} = \frac{r_k}{\max\{\max_i\{\|A_i u_{i,k}\|_2\}, \|b\|_2\}} \quad \text{and} \quad d_{i,k}^{norm} = \frac{d_{i,k}}{\min_i\{\|A_i^T \lambda_k\|_2\}}, \tag{5.2}$$

the normalized residual balancing schema with respect to r_k^{norm}, d_k^{norm} is

$$\tau_{k+1} = \begin{cases} \eta\tau_k & \text{if} \quad \max_i \|d_{i,k}^{norm}\|_2 > \mu\|r_k^{norm}\|_2 \\ \tau_k/\eta & \text{if} \quad \|r_k^{norm}\|_2 > \mu \max_i \|d_{i,k}^{norm}\|_2 \\ \tau_k & \text{otherwise,} \end{cases} \tag{5.3}$$

where $\mu > 1$ and $\eta > 1$ are parameters.

5.1.2 Dual interpretation of multi-block ADMM

The dual of multi-block problem (2.8) has the form

$$\min_{\zeta\in\mathbb{R}^p} \underbrace{H_1^*(A_1^T\zeta) - \langle\zeta, b\rangle}_{\hat{H}_1(\zeta)} + \sum_{i=2}^{N} \underbrace{H_i^*(A_i^T\zeta)}_{\hat{H}_i(\zeta)}; \tag{5.4}$$

with F^* denoting the Fenchel conjugate of F, defined as $F^*(y) = \sup_x \langle x, y\rangle - F(x)$ [Roc70].

The direct extension of DRS algorithm to multi-block problem solves (5.4) by generating sequences $(\zeta_{i,k})_{i\in\mathbb{N},k\in\mathbb{N}}$ according to

$$0 \quad \in \quad \frac{\zeta_{i,k+1} - \zeta_k}{\tau_k} + \sum_{j=1}^{i} \partial\hat{H}_j(\zeta_{i,k+1}) + \sum_{j=i+1}^{N} \partial\hat{H}_j(\zeta_k), \tag{5.5}$$

where $\zeta_k = \zeta_{N,k}$,and we use the standard notation $\partial F(x)$ for the subdifferential of F evaluated at x [Roc70].

Similar to the two-block ADMM, referring back to (2.9) and defining

$$\lambda_{i,k+1} = \lambda_k + \tau_k(b - \sum_{j=1}^{i} A_i u_{i,k+1} - \sum_{j=i+1}^{N} A_i u_{i,k}), \tag{5.6}$$

the optimality condition for the minimization of (2.9) is

$$0 \in \partial H_i(u_{i,k+1}) - A_i^T\lambda_{i,k+1}, \tag{5.7}$$

which is equivalent to $A_i^T\lambda_{i,k+1} \in \partial H_i(u_{i,k+1})$, thus $u_{i,k+1} \in \partial H_i^*(A_i^T\lambda_{i,k+1})$. Recall-

ing definition of $\hat{H}_i(\lambda)$ in (5.4), we arrive at

$$A_1 u_1^{k+1} - b \in \partial \hat{H}_1(\lambda_1^{k+1}) \quad \text{and} \quad A_i u_{i,k+1} \in \partial \hat{H}_i(\lambda_{i,k+1}), \quad 2 \leq i \leq N. \tag{5.8}$$

Using these identities, we finally have

$$\lambda_{i,k+1} = \lambda_k + \tau_k (b - \sum_{j=1}^{i} A_i u_{i,k+1} - \sum_{j=i+1}^{N} A_i u_{i,k}) \tag{5.9}$$

$$\in \lambda_k - \tau_k \big(\sum_{j=1}^{i} \partial \hat{H}_i(\zeta_{i,k+1}) + \sum_{j=i+1}^{N} \partial \hat{H}_i(\zeta_{i,k}) \big) \tag{5.10}$$

showing that the sequences $(\lambda_{i,k})_{i \in \mathbb{N}, k \in \mathbb{N}}$ satisfy the multi-block DRS conditions (5.5) as $(\zeta_{i,k})_{i \in \mathbb{N}, k \in \mathbb{N}}$.

5.1.3 Spectral stepsize for multi-block DRS

Similar to the two-block analysis in Section 4.2.1, considering the dual problem (5.4), $\partial \hat{H}_i(\zeta_i)$ at iteration k is modeled by linear functions of their arguments,

$$\partial \hat{H}_i(\zeta_i) = \alpha_{i,k} \, \zeta_i + \Psi_{i,k} \tag{5.11}$$

where $\alpha_{i,k} > 0$ are local curvature estimates of functions $\hat{H}_i$, respectively, and $\Psi_{i,k} \subset \mathbb{R}^p$. Once we obtain these curvature estimates, we will be able to exploit the following propositions.

Proposition 5.1.1 (Spectral multi-block DRS)**.** *Suppose the multi-block DRS steps* (5.5) *are applied to problem* (5.4)*, where (omitting iteration* k *from* $\alpha_{i,k}, \Psi_{i,k}$ *to*

lighten the notation in what follows)

$$\partial \hat{H}_i(\zeta_i) = \alpha_i \, \zeta_i + \Psi_i$$

Then, the minimal residual of $\sum_{i=1}^{N} \hat{H}_i(\zeta_{i,k+1})$ *is obtained by setting*

$$\tau_k = \text{argmin}_\tau \sum_{i=1}^{N} \log(1 + \alpha_i \, \tau) - \log \tau.$$

Proof. Inserting (5.11) into the DRS step (5.5), we have

$$0 \;\in\; \frac{\zeta_{i,k+1} - \zeta_k}{\tau} + \sum_{j=1}^{i} (\alpha_j \, \zeta_{j,k+1} + \Psi_j) + \sum_{j=i+1}^{N} (\alpha_j \, \zeta_{j,k} + \Psi_j) \tag{5.12}$$

From (5.12), we can explicitly get the update for $\zeta_{i,k+1}$ as

$$\begin{aligned}
\zeta_{1,k+1} &= \frac{1 + \alpha_1 \, \tau - \alpha \, \tau}{1 + \alpha_1 \, \tau} \zeta_k - \frac{a \, \tau}{1 + \alpha_1 \, \tau} && (5.13)\\
\zeta_{i,k+1} &= \frac{1}{1 + \alpha_i \, \tau} \zeta_{i-1,k+1} + \frac{\alpha_i \, \tau}{1 + \alpha_i \, \tau} \zeta_k && (5.14)\\
&= \frac{1}{\prod_{j=2}^{i} (1 + \alpha_j \, \tau)} \zeta_{1,k+1} + \sum_{j=2}^{i} \frac{\alpha_j \, \tau}{\prod_{l=j}^{i} (1 + \alpha_l \, \tau)} \zeta_k && (5.15)\\
&= \frac{1 + \alpha_1 \, \tau - \alpha \, \tau}{\prod_{j=1}^{i} (1 + \alpha_j \, \tau)} \zeta_k - \frac{a \, \tau}{\prod_{j=1}^{i} (1 + \alpha_j \, \tau)} + \sum_{j=2}^{i} \frac{\alpha_j \, \tau}{\prod_{l=j}^{i} (1 + \alpha_l \, \tau)} \zeta_k && (5.16)\\
&= \frac{1}{\prod_{j=1}^{i} (1 + \alpha_j \, \tau)} \left((1 - \alpha \, \tau) \, \zeta_k + \sum_{l=1}^{i} \alpha_l \, \tau \prod_{m=1}^{l-1} (1 + \alpha_m \, \tau) \, \zeta_k - a \, \tau \right) && (5.17)\\
&= \zeta_k - \frac{\alpha \, \tau \, \zeta_k + a \, \tau}{\prod_{j=1}^{i} (1 + \alpha_j \, \tau)}, && (5.18)
\end{aligned}$$

where $a_i \in \Psi_i$, $a = \sum_{i=1}^N a_i$ and $\alpha = \sum_{i=1}^N \alpha_i$, and $\zeta_{k+1} = \zeta_{N,k+1}$.

The residual r_{DR} at ζ_{k+1} is simply the magnitude of the subgradient of the objective that is given by

$$r_{DR} = \|\alpha\zeta_{k+1} + a\|_2 = \left(1 - \frac{\alpha\,\tau}{\prod_{j=1}^N (1+\alpha_j\,\tau)}\right) \|\alpha\zeta_k + a\|_2, \tag{5.19}$$

where ζ_{k+1} in (5.19) was substituted with (5.18) when $i = N$. The optimal stepsize τ_k minimizes the residual

$$\begin{aligned}
\tau_k &= \arg\min_\tau r_{\text{DR}} = \arg\max_\tau \frac{\alpha\,\tau}{\prod_{j=1}^N (1+\alpha_j\,\tau)} && (5.20)\\
&= \arg\min_\tau -\log\left(\frac{\alpha\,\tau}{\prod_{j=1}^N (1+\alpha_j\,\tau)}\right) && (5.21)\\
&= \text{argmin}_\tau \sum_{i=1}^N \log(1+\alpha_i\,\tau) - \log\tau && (5.22)\\
&= \text{argmin}_\tau \sum_{i=1}^N \log(\hat{\alpha}_i + \tau) - \log\tau, && (5.23)
\end{aligned}$$

where $\hat{\alpha}_i = 1/\alpha_i$ is the spectral gradient descent stepsizes for $\hat{H}_i$ at $\zeta_{i,k}$. □

When $N = 2$, spectral stepsize for multi-block DRS in (5.23) reduce to (4.20) in Section 4.2.1. (5.23) is an one-dimensional optimization problem that can be solved by gradient method such as BFGS and L-BFGS [Byr+95]. Gradient method will find a local minimum that is closest to the initial stepsize if (5.23) is nonconvex. Inspired by the closed-form solution for spectral stepsize of DRS in (4.20), the geometric mean of curvatures of functions is used to estimate the solution of (5.23)

as spectral stepsize,

$$\tau_k = \left(\prod_{i=1}^{N} \hat{\alpha}_i \right)^{\frac{1}{N}}. \tag{5.24}$$

The following proposition is exploited as an substitution of stepsize selection for multi-block DRS.

Proposition 5.1.2 (Approximate spectral multi-block DRS)**.** *Suppose the multi-block DRS steps* (5.5) *are applied to problem* (5.4)*, where (omitting iteration k from $\alpha_{i,k}, \Psi_{i,k}$ to lighten the notation in what follows)*

$$\partial \hat{H}_i(\zeta_i) = \alpha_i \zeta_i + \Psi_i$$

Then, the spectral stepsize can be estimated by $\tau_k = \left(\prod_{i=1}^{N} \frac{1}{\alpha_i} \right)^{\frac{1}{N}}$.

5.1.4 Spectral penalty parameter for multi-block ADMM

The spectral penalty parameter selection algorithm is introduced in this section. Similar to Algorithm 1 for two-block ADMM, we can begin by obtaining linear estimates of the subgradients of the terms in the dual objective (5.4). Thanks to the equivalence of ADMM and DRS presented in Section 5.1.2, the subgradients can be estimated along with ADMM steps. The correlation based safeguarding criterion is applied to each curvature estimation. Then Proposition 5.1.1 and Proposition 5.1.2 show how to adaptively choose the stepsize for mutli-block DRS, i.e. the penalty parameter for multi-block ADMM.

Defining

$$\Delta\lambda_{i,k} := \lambda_{i,k} - \lambda_{i,k_0} \quad \text{and} \quad \Delta\hat{H}_{i,k} := \partial\hat{H}_i(\lambda_{i,k}) - \partial\hat{H}_i(\lambda_{i,k_0}) = A(u_{i,k} - u_{i,k_0}), \tag{5.25}$$

the spectral stepsize $\alpha_{i,k} = 1/\alpha_k$ for the components $\hat{H}(\lambda_{i,k})$ at the k-th iteration are estimated by

$$\hat{\alpha}^{\text{SD}}_{i,k} = \frac{\langle \Delta\lambda_{i,k}, \Delta\lambda_{i,k} \rangle}{\langle \Delta\hat{H}_{i,k}, \Delta\lambda_{i,k} \rangle} \qquad \text{and} \qquad \hat{\alpha}^{\text{MG}}_{i,k} = \frac{\langle \Delta\hat{H}_{i,k}, \Delta\lambda_{i,k} \rangle}{\langle \Delta\hat{H}_{i,k}, \Delta\hat{H}_{i,k} \rangle} \tag{5.26}$$

$$\hat{\alpha}_{i,k} = \begin{cases} \hat{\alpha}^{\text{MG}}_{i,k} & \text{if } \; 2\,\hat{\alpha}^{\text{MG}}_{i,k} > \hat{\alpha}^{\text{SD}}_{i,k} \\ \hat{\alpha}^{\text{SD}}_{i,k} - \hat{\alpha}^{\text{MG}}_{i,k}/2 & \text{otherwise.} \end{cases} \tag{5.27}$$

The correlation to test the validity of the linear assumption (5.11) for each component is measured by

$$\alpha^{\text{corr}}_{i,k} = \frac{\langle \Delta\hat{H}_{i,k}, \Delta\lambda_{i,k} \rangle}{\|\Delta\hat{H}_{i,k}\|_2 \; \|\Delta\lambda_{i,k}\|_2} \tag{5.28}$$

Proposition 5.1.3 (Safeguarding)**.** *The threshold ϵ^{corr} on $\alpha^{corr}_{i,k}$ is used to safeguard the curvature estimations. If none of the curvatures is deemed accurate, i.e. $\forall i \in \{1 \dots N\}$, $\alpha^{corr}_{i,k} < \epsilon^{corr}$, then $\tau_{k+1} = \tau_k$. Otherwise, the inaccurate stepsizes (curvatures) are estimated by the largest stepsizes (smallest curvatures) among the accurate estimations, $\forall i \in \{j | \alpha^{corr}_{j,k} < \epsilon^{corr}\}, \hat{\alpha}_{i,k} = \max\{\hat{\alpha}_{j,k} | \alpha^{corr}_{j,k} \geq \epsilon^{corr}\}$.*

Then the spectral penalty parameter is selected by either (5.23) or (5.24).

Algorithm 2 Adaptive multi-block ADMM with spectral penalty parameter selection

Input: initialize v_0, λ_0, τ_0, $k_0 = 0$

while not converge by residual check (2.17) **and** $k <$ maxiter **do**
  for $i = 1, 2, \ldots, N$ **do**
    $u_{i,k+1} \leftarrow \arg\min_{u_i} H_i(u_i) + \langle \lambda_k, -A_i u_i \rangle + \frac{\tau_k}{2} \|b - \sum_{j=1, j\neq i}^{N} A_j u_{j,k+1} - A_i u_i\|_2^2$
  end for
  $\lambda_{k+1} \leftarrow \lambda_k + \tau_k (b - \sum_{i=1}^{N} A_i u_{i,k+1})$
  if $\mathrm{mod}(k, T_f) = 1$ **then**
    for $i = 1, 2, \ldots, N$ **do**
      $\lambda_{i,k+1} \leftarrow \lambda_k + \tau_k (b - \sum_{j=1}^{i} A_i u_{i,k+1} - \sum_{j=i+1}^{N} A_i u_{i,k})$
      Compute spectral stepsizes $\hat{\alpha}_{i,k+1}$ using (5.27)
      Estimate correlation $\alpha^{\mathrm{corr}}_{i,k+1}$ using (5.28)
    end for
    Safeguard $\hat{\alpha}_{i,k+1}$ based on $\alpha^{\mathrm{corr}}_{i,k+1}$ using Proposition 5.1.3
    Update τ_{k+1} using either (5.23) or (5.24)
    $k_0 \leftarrow k$
  else
    $\tau_{k+1} \leftarrow \tau_k$
  end if
end while

The proposed method for spectral penalty parameter selection falls back to (5.151) when $N = 2$. The complete *adaptive multi-block ADMM* (AADMM) is shown in Algorithm 2.

5.1.5 Experiment: elastic net regularized linear regression

Solving EN regularized linear regression problem (2.35) with two-block ADMM is presented in the previous sections. An alternative approach is to apply multi-block ADMM by solving

$$\begin{aligned} \min_{u_1,u_2,u_3} \ & \frac{1}{2}\|Du_1 - c\|_2^2 + \rho_1\|u_2\|_1 + \frac{\rho_2}{2}\|u_3\|_2^2 \\ \text{subject to} \ & u_1 - u_2 = 0,\ u_1 - u_3 = 0. \end{aligned} \tag{5.29}$$

Then the ADMM steps are

$$u_{1,k+1} = \arg\min_{u_1} \frac{1}{2}\|Du_1 - c\|_2^2 + \frac{\tau}{2}\sum_{j=1}^{2}\|u_{j+1,k} - u_1 + \lambda_{j,k}/\tau\|_2^2 \tag{5.30}$$

$$= \begin{cases} (D^TD + 2\tau I_n)^{-1}(\tau(u_{2,k} + u_{3,k}) + (\lambda_{1,k} + \lambda_{2,k}) + D^Tc) & \text{if } n \geq m \\ (\frac{1}{2}I_n - D^T(4\tau I_m + 2DD^T)^{-1}D)((u_{2,k} + u_{3,k}) + (\lambda_{1,k} + \lambda_{2,k})/\tau + D^Tc/\tau) & \text{if } n < m \end{cases} \tag{5.31}$$

$$u_{2,k+1} = \arg\min_{u_2} \rho_1\|u_2\|_1 + \frac{\tau}{2}\| - u_{1,k+1} + u_2 + \lambda_{1,k}/\tau\|_2^2 \tag{5.32}$$

$$= \text{shrink}(u_{1,k+1} - \lambda_{1,k}, \rho_1/\tau) \tag{5.33}$$

$$u_{3,k+1} = \arg\min_{u_3} \frac{\rho_2}{2}\|u_3\|_2^2 + \frac{\tau}{2}\| - u_{k+1} + u_3 + \lambda_{2,k}/\tau\|_2^2 \tag{5.34}$$

$$= \frac{1}{\rho_2 + \tau}(\tau u_1 - \lambda_2) \tag{5.35}$$

$$\lambda_{1,k+1} = \lambda_{1,k} + \tau(-u_{1,k+1} + u_{2,k+1}) \tag{5.36}$$

$$\lambda_{2,k+1} = \lambda_{2,k} + \tau(-u_{1,k+1} + u_{3,k+1}). \tag{5.37}$$

Dataset	Vanilla ADMM	Residual balance	Normalized RB	Approx AADMM	**Adaptive ADMM**
Synthetic	2000+(1.64)	102(.103)	147(.144)	116(.122)	**62(.167)**
Boston	2000+(1.44)	48(.032)	81(.056)	**21(.020)**	**63(.781)**
Diabetes	526(.509)	17(.019)	16(.018)	**12(.019)**	**33(.393)**
Leukemia	2000+(9.06)	1094(4.84)	**78(.331)**	1000(4.69)	**1096(5.13)**
Prostate	514(.314)	27(.017)	24(.018)	**19(.019)**	**19(.078)**
Servo	116(.089)	16(.012)	16(.017)	16(.016)	**13(.051)**

Table 5.1: Iterations (and runtime in seconds) for EN regularized linear regression. Absence of convergence after n iterations is indicated as $n+$. Approx AADMM and Adaptive ADMM are the proposed Algorithm 2 with (5.23) and (5.24), respectively.

Convergence results Both synthetic data and regression benchmark data are investigated. Typical parameters $\rho_1 = \rho_2 = 1$ are used in all experiments. Table 5.1 reports the convergence speed of ADMM and its variants for applying multi-block

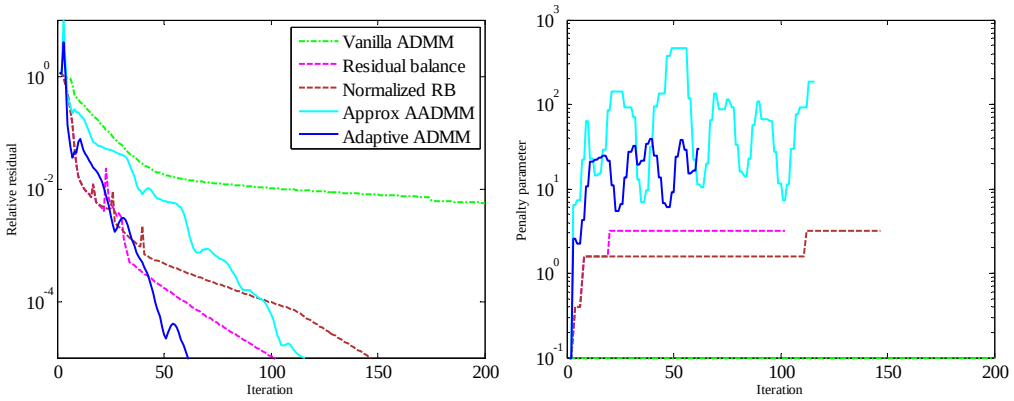


Figure 5.1: Relative residual (left) and penalty parameter (right) for applying multi-block ADMM to the synthetic problem of EN regularized linear regression.

ADMM to EN regularized linear regression. Comparing with two-block ADMM results in Section 4.3, multi-block ADMM often requires more iterations to converge. Both Multi-block AADMM by optimize (5.23) and Approx AADMM by using an approximate solution in (5.24) requires less iterations than vanilla ADMM. However, the runtime of Approx AADMM is notably shorter than AADMM since the formulation (5.24) is much simpler. Multi-block residual balancing methods proposed in Section 5.1.1 also perform well, and sometimes outperforms AADMM.

The convergence curve and the adapted penalty parameter for applying multi-block ADMM to the synthetic problem of EN regularized linear regression are presented in Fig. 5.1. Both AADMM methods and residual balancing methods increase the initial penalty parameter.

Sensitivity Fig. 5.2 presents iteration counts for a wide range of values of initial penalty parameter τ_0 and problem scale s. Scaling sensitivity experiments were done by multiplying the groundtruth vector x^* by a scalar s in the synthetic problem. Both AADMM methods and residual balancing methods are relatively stable with

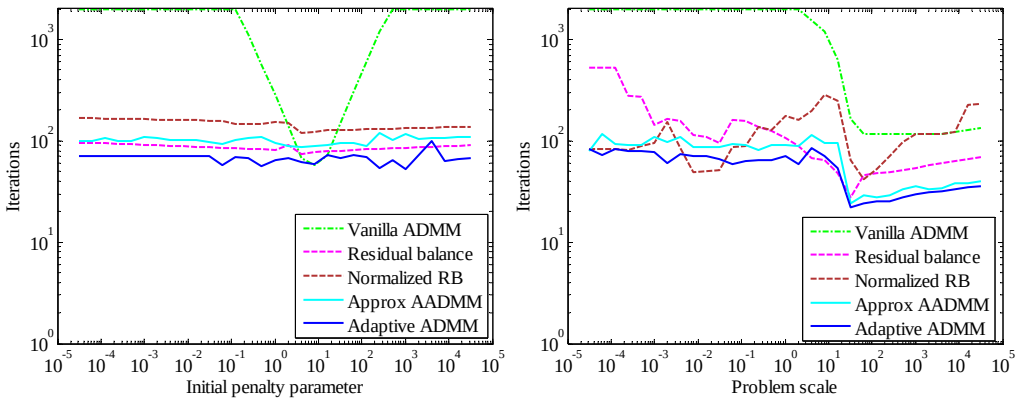


Figure 5.2: Sensitivity with respect to initial penalty parameter (left) and problem scale (right) for applying multi-block ADMM to the synthetic problem of EN regularized linear regression.

respect to the selection of initial penalty parameter τ_0. AADMM methods are relatively more stable than residual balancing methods with respect to the problem scale s.

5.1.6 Experiment: robust principal component analysis

We consider the stable version of RPCA [Zho+10; TY11]

$$\min_{A,E,Z} \|A\|_* + \rho_1\|E\|_1 + \frac{\rho_2}{2}\|Z\|_F^2 \quad \text{subject to} \quad A + E + Z = C, \tag{5.38}$$

where A represents the low rank matrix, E represents the sparse error, Z represents the noise. Multi-block ADMM is applied to RPCA by iteratively solving Z, E and

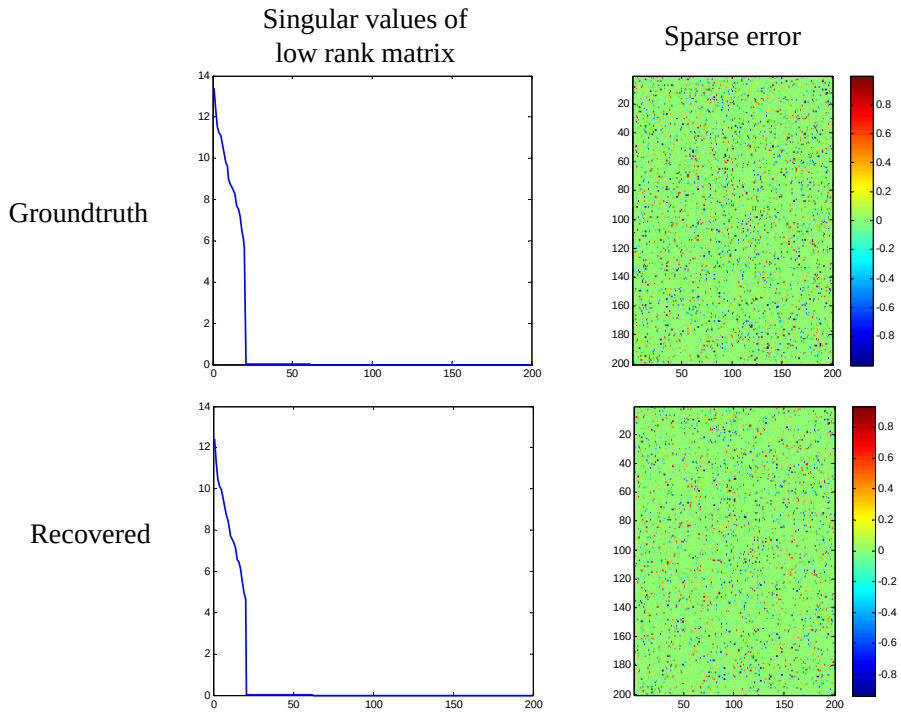


Figure 5.3: Singular values of low rank matrix A (left) and sparse error E (right) for the synthetic problem of RPCA. The bottom row is recovered by AADMM, where mean square errors of recovered A and E are $1.48\text{e}-4$ and $1.88\text{e}-4$.

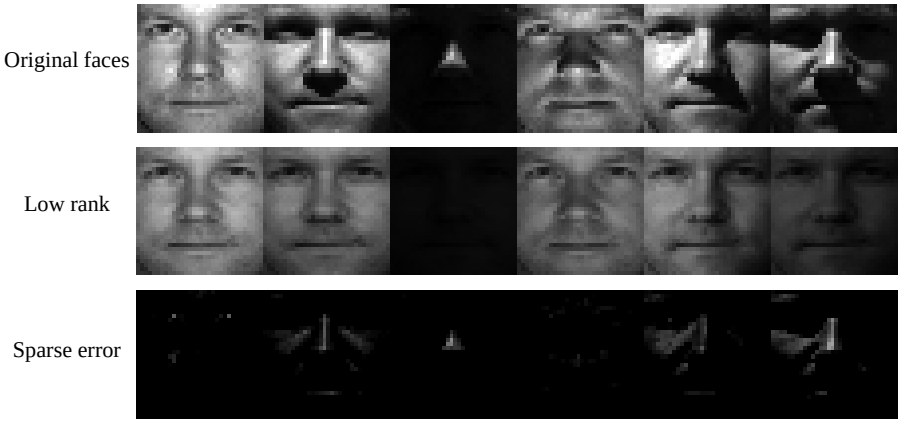


Figure 5.4: Sample face images of human subject 3 and recovered low rank faces and sparse errors by AADMM. RPCA decomposes the original faces into intrinsic images (low rank) and shadings (sparse).

A,

$$
\begin{aligned}
Z_{k+1} &= \frac{\rho_2}{2}\|Z\|_F^2 + \frac{\tau_k}{2}\|C - E_k - A_k - Z + \lambda_k/\tau_k\|_F^2 && (5.39)\\
&= \frac{\tau_k}{\rho_2 + \tau_k}(C - E_k - A_k - Z + \lambda_k/\tau_k) && (5.40)\\
E_{k+1} &= \rho_1\|E\|_1 + \frac{\tau_k}{2}\|C - E - A_k - Z_{k+1} + \lambda_k/\tau_k\|_F^2 && (5.41)\\
&= \text{shrink}(C - A_k - Z_{k+1} + \lambda_k/\tau_k, \rho_1/\tau_k) && (5.42)\\
A_{k+1} &= \|A\|_* + \frac{\tau_k}{2}\|C - E_{k+1} - A - Z_{k+1} + \lambda_k/\tau_k\|_F^2 && (5.43)\\
&= \text{SVT}(C - E_{k+1} - Z_{k+1} + \lambda_k/\tau_k,\, 1/\tau_k) && (5.44)\\
\lambda_{k+1} &= \lambda_k + \tau_k(C - E_{k+1} - A_{k+1} - Z_{k+1}). && (5.45)
\end{aligned}
$$

A synthetic problem is created as follows, let low rank matrix $A = UV$, where $U \in \mathbb{R}^{n\times r}$, $V \in \mathbb{R}^{r\times n}$ are two random matrix with $n = 200$ and $r = 20$. A is then normalized to have each entry between -1 and 1. E is a sparse matrix of 5% entries as 1 and 5% entries as -1. Z is Gaussian noise with zero mean and 0.01 standard deviation. Extended Yale B Face dataset is used by applying RPCA for each individual human, and the measurement matrix C is constructed by vectorizing each image as a row of the matrix, which is composed of a low rank matrix, a sparse error matrix and some noise. $\rho_2 = 1$, $\rho_1 = \rho_2/10$ are used for the synthetic problem, and $\rho_2 = 0.1$, $\rho_1 = \rho_2/5$ are used for face decomposition. Those parameters are selected based on the performance of reconstruction. The synthetic problem are presented in Fig. 5.3, and example face images are presented in Fig. 5.4.

Dataset	Vanilla ADMM	Residual balance	Normalized RB	Approx AADMM	**Adaptive ADMM**
Synthetic	42(1.07)	18(.449)	36(.870)	18(.476)	**13(.412)**
Hum1	20(.594)	20(.532)	26(.710)	19(.584)	**16(.582)**
Hum2	18(.505)	18(.470)	27(.718)	18(.545)	**15(.548)**
Hum3	20(.532)	20(.498)	26(.643)	19(.515)	**16(.530)**

Table 5.2: Iterations (and runtime in seconds) for robust PCA. Absence of convergence after n iterations is indicated as $n+$. Approx AADMM and Adaptive ADMM are the proposed Algorithm 2 with (5.23) and (5.24), respectively.

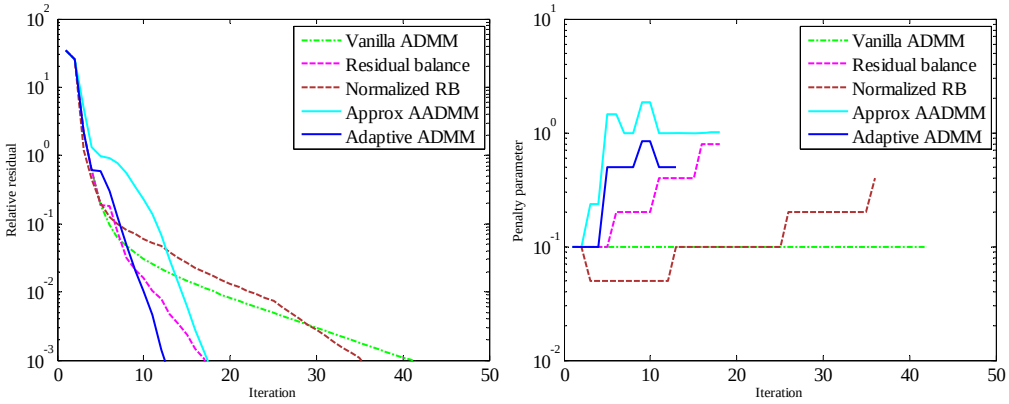


Figure 5.5: Relative residual (left) and penalty parameter (right) for the synthetic problem of RPCA.

Convergence results Table 5.2 reports the convergence speed of ADMM and its variants for RPCA. Multi-block AADMM by optimizing (5.23) often requires less iteration counts than Approx AADMM by using an approximate solution in (5.24). However, the runtime of Approx AADMM is somtimes shorter than AADMM since the formulation (5.24) is simpler. The initial penalty parameter choice $\tau_0 = 0.1$ works well for vanilla ADMM when applying RPCA to faces. The convergence curve and the adapted penalty parameter for applying multi-block ADMM to the synthetic problem of RPCA are presented in Fig. 5.5.

Sensitivity Fig. 5.6 presents iteration counts for a wide range of values of initial penalty parameter τ_0 and problem scale s. Scaling sensitivity experiments were done by multiplying the measurement C by a scalar s in the synthetic problem. Both AADMM methods and residual balancing methods are stable with respect to the selection of initial penalty parameter τ_0. Normalized residual balancing performs better when problem scale s is very small or large, while AADMM performs better for the normalized data in synthetic problem when s is close to 1.

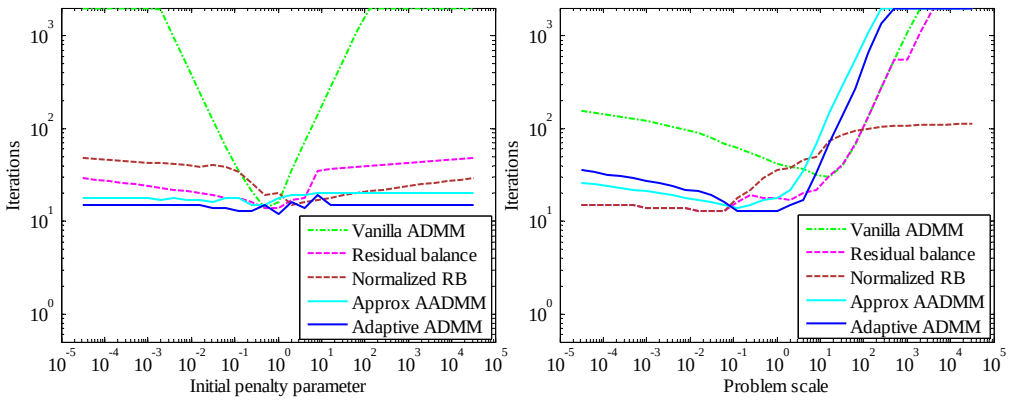


Figure 5.6: Sensitivity with respect to initial penalty parameter (left) and problem scale (right) for applying multi-block ADMM to the synthetic problem of EN regularized linear regression.

5.2 Adaptive Relaxed ADMM

5.2.1 Introduction

Relaxed ADMM is a popular practical variant of ADMM, and proceeds with the following steps:

$$u_{k+1} = \arg\min_u h(u) + \frac{\tau_k}{2}\left\|b - Au - Bv_k + \frac{\lambda_k}{\tau_k}\right\|^2 \tag{5.46}$$

$$\hat{u}_{k+1} = \gamma_k A u_{k+1} + (1-\gamma_k)(b - Bv_k) \tag{5.47}$$

$$v_{k+1} = \arg\min_v g(v) + \frac{\tau_k}{2}\left\|b - \hat{u}_{k+1} - Bv + \frac{\lambda_k}{\tau_k}\right\|^2 \tag{5.48}$$

$$\lambda_{k+1} = \lambda_k + \tau_k(b - \hat{u}_{k+1} - Bv_{k+1}). \tag{5.49}$$

Here, $\lambda_k \in \mathbb{R}^p$ denotes the dual variables (Lagrange multipliers) on iteration k, and (τ_k, γ_k) are sequences of penalty and relaxation parameters. Relaxed ADMM coincides with the original non-relaxed version if $\gamma_k = 1$.

Convergence of (relaxed) ADMM is guaranteed under fairly general assump-

tions [EB92; HY12b; HY15; Fan+15], if the penalty and relaxation parameters are held constant. However, the practical performance of ADMM depends strongly on the choice of these parameters, as well as on the problem being solved. Good penalty choices are known for certain ADMM formulations, such as strictly convex quadratic problems [RDC14; Gha+15], and for the gradient descent parameter in the "linearized" ADMM [Lin+11; Liu+13].

Adaptive penalty methods (in which the penalty parameters are tuned automatically as the algorithm proceeds) achieve good performance without user oversight. For non-relaxed ADMM, the authors of [He+00] propose methods that modulate the penalty parameter so that the primal and dual residuals (i.e., derivatives of the Lagrangian with respect to primal and dual variables) are of approximately equal size. This "residual balancing" approach has been generalized to work with preconditioned variants of ADMM [Gol+15] and distributed ADMM [Son+16]. In [Xu+17b], a spectral penalty parameter method is proposed that uses the local curvature of the objective to achieve fast convergence. All of these methods are specific to (non-relaxed) *vanilla* ADMM, and do not apply to the more general case involving a relaxation parameter.

In this section, we study adaptive parameter choices for the relaxed ADMM that jointly and automatically tune both the penalty parameter τ_k and relaxation parameter γ_k. In Section 5.2.2, we address theoretical questions about the convergence of ADMM with non-constant penalty and relaxation parameters. In Section 5.2.4, we discuss practical methods for choosing these parameters. Finally, in Section 5.2.7, we compare ARADMM to other ADMM variants and examine the

benefits of the proposed approach for real-world regression, classification, and image processing problems.

5.2.2 Convergence theory

We study conditions under which ADMM converges with adaptive penalty and relaxation parameters. Our approach utilizes the variational inequality (VI) methods put forward in [He+00; HY12b; HY15]. Our results measure convergence using the primal and dual "residuals," which are defined as

$$r_k = b - Au_k - Bv_k \text{ and } d_k = \tau_k A^T B(v_k - v_{k-1}). \tag{5.50}$$

It has been observed that these residuals approach zero as the algorithm approaches a true solution [Boy+11]. Typically, the iterative process is stopped if

$$\begin{aligned} &\|r_k\| \leq \epsilon^{tol} \max\{\|Au_k\|, \|Bv_k\|, \|b\|\} \\ &\text{and} \quad \|d_k\| \leq \epsilon^{tol} \|A^T \lambda_k\|, \end{aligned} \tag{5.51}$$

where $\epsilon^{tol} > 0$ is the stopping tolerance [Boy+11]. For this reason, it is important to know that the method converges in the sense that the residuals approach zero as $k \to \infty$.

In the sequel, we prove that relaxed ADMM converges in the residual sense, provided that the algorithm parameters satisfy one of the following two assumptions.

Assumption 5.2.1. *The relaxation sequence γ_k and penalty sequence τ_k satisfy*

$$\begin{aligned} &1 \leq \gamma_k < 2, \ \lim_{k\to\infty} 1/\tau_k^2 < \infty, \ \sum_{k=1}^{\infty} \eta_k^2 < \infty, \\ &\textit{where} \quad \eta_k^2 = \frac{\gamma_k}{(2-\gamma_k)} \max\left(\tau_k^2/\tau_{k-1}^2, \, 1\right) - 1. \end{aligned} \tag{5.52}$$

Assumption 5.2.2. *The relaxation sequence γ_k and penalty sequence τ_k satisfy*

$$\begin{aligned} &1 \leq \gamma_k < 2, \ \lim_{k\to\infty} \tau_k^2 < \infty, \ \sum_{k=1}^{\infty} \theta_k^2 < \infty, \\ &\textit{where} \quad \theta_k^2 = \frac{\gamma_k}{(2-\gamma_k)} \max\left(\tau_{k-1}^2/\tau_k^2, \, 1\right) - 1. \end{aligned} \tag{5.53}$$

In Section 5.2.5, we prove adaptive relaxed ADMM converges if the algorithm parameters satisfy either Assumption 5.2.1 or Assumption 5.2.2. Before presenting the proof, we show how to choose the relaxation parameters that lead to efficient performance in practice.

5.2.3 Dual interpretation of relaxed ADMM

We derive our adaptive stepsize rules by examining the close relationship between relaxed ADMM and the relaxed Douglas-Rachford Splitting (DRS) [EB92; DY14; GB16]. The dual of the general constrained problem (5.156) is

$$\min_{\zeta\in\mathbb{R}^p} \underbrace{h^*(A^T\zeta) - \langle \zeta, b\rangle}_{\hat{h}(\zeta)} + \underbrace{g^*(B^T\zeta)}_{\hat{g}(\zeta)}, \tag{5.54}$$

with f^* denoting the Fenchel conjugate of f, defined as $f^*(y) = \sup_x \langle x, y\rangle - f(x)$ [Roc70].

The relaxed DRS algorithm solves (5.54) by generating two sequences, $(\zeta_k)_{k\in\mathbb{N}}$ and $(\hat{\zeta}_k)_{k\in\mathbb{N}}$, according to

$$0 \in \frac{\hat{\zeta}_{k+1} - \zeta_k}{\tau_k} + \partial\hat{h}(\hat{\zeta}_{k+1}) + \partial\hat{g}(\zeta_k), \tag{5.55}$$

$$0 \in \frac{\zeta_{k+1} - \zeta_k}{\tau_k} + \gamma_k\,\partial\hat{h}(\hat{\zeta}_{k+1}) - (1-\gamma_k)\partial\hat{g}(\zeta_k) + \partial\hat{g}(\zeta_{k+1}), \tag{5.56}$$

where γ_k is a relaxation parameter, and $\partial f(x)$ denotes the subdifferential of f evaluated at x [Roc70]. Referring back to ADMM in (5.46)–(5.49), and defining $\hat{\lambda}_{k+1} = \lambda_k + \tau_k(b - Au_{k+1} - Bv_k)$, the sequences $(\lambda_k)_{k\in\mathbb{N}}$ and $(\hat{\lambda}_k)_{k\in\mathbb{N}}$ satisfy the same conditions (5.55) and (5.56) as $(\zeta_k)_{k\in\mathbb{N}}$ and $(\hat{\zeta}_k)_{k\in\mathbb{N}}$, thus ADMM for the problem (5.156) is equivalent to DRS on its dual (5.54). A detailed proof of this is provided in the supplementary material.

5.2.4 Spectral adaptive stepsize rule

Adaptive stepsize rules of the "spectral" type were originally proposed for simple gradient descent on smooth problems by Barzilai and Borwein [BB88], and have been found to dramatically outperform constant stepsizes in many applications [Fle05; Wri+09c]. Spectral stepsize methods work by modeling the gradient of the objective as a linear function, and then selecting the optimal stepsize for this simplified linear model.

Spectral methods were recently used to determine the penalty parameter for the non-relaxed ADMM in [Xu+17b]. Inspired by that work, we derive spectral stepsize rules assuming a linear model/approximation for $\partial\hat{h}(\hat{\zeta})$ and $\partial\hat{g}(\zeta)$ at iteration

k given by

$$\partial\hat{h}(\hat{\zeta}) = \alpha_k\,\hat{\zeta} + \Psi_k \qquad \text{and} \qquad \partial\hat{g}(\zeta) = \beta_k\,\zeta + \Phi_k, \tag{5.57}$$

where $\alpha_k > 0$, $\beta_k > 0$ are local curvature estimates of $\hat{h}$ and $\hat{g}$, respectively, and $\Psi_k, \Phi_k \subset \mathbb{R}^p$. Once we obtain these curvature estimates, we will exploit the following simple proposition whose proof is given in the supplementary material.

Proposition 5.2.1. *Suppose the DRS steps* (5.55)–(5.56) *are applied to problem* (5.54)*, where (omitting iteration k from $\alpha_k, \beta_k, \Psi_k, \Phi_k$ to lighten the notation in what follows)*

$$\partial\hat{h}(\hat{\zeta}) = \alpha\,\hat{\zeta} + \Psi \qquad \textit{and} \qquad \partial\hat{g}(\zeta) = \beta\,\zeta + \Phi. \tag{5.58}$$

Then, the residual of $\hat{h}(\zeta_{k+1}) + \hat{g}(\zeta_{k+1})$ will be zero if τ and γ are chosen to satisfiy $\gamma_k = 1 + \frac{1+\alpha\beta\tau_k^2}{(\alpha+\beta)\tau_k}$.

Our adaptive method works by fitting a linear model to the gradient (or subgradient) of our objective, and then using Proposition 5.2.1 to select an optimal stepsize pair that obtains zero residual on the model problem. For our convergence theory to hold, we need $\gamma < 2$. For fixed values of α and β, the minimal value of γ_k that is still optimal for the linear model occurs if we choose

$$\tau_k = \arg\min_{\tau} \frac{1 + \alpha\beta\tau^2}{(\alpha+\beta)\tau} = 1/\sqrt{\alpha\beta}. \tag{5.59}$$

Note this is the same "optimal" penalty parameter proposed for non-relaxed ADMM in [Xu+17b]. Under this choice of τ_k, we then have the "optimal" relaxation param-

eter

$$\gamma_k = 1 + \frac{1 + \alpha\beta\tau^2}{(\alpha+\beta)\tau} = 1 + \frac{2\sqrt{\alpha\beta}}{\alpha+\beta} \leq 2. \tag{5.60}$$

Estimation of stepsizes We now propose a simple method for fitting a linear model to the dual objective terms so that the formulas in Section **??** can be used to obtain stepsizes. Once these linear models are formed, the optimal penalty parameter and relaxation term can be calculated by (5.59) and (5.60), thanks to the equivalence of relaxed ADMM and DRS.

In what follows, we let $\hat{\alpha}_k = 1/\alpha_k$ and $\hat{\beta}_k = 1/\beta_k$ to simplify notation. The optimal stepsize choice is then written as $\tau_k = (\hat{\alpha}_k\,\hat{\beta}_k)^{1/2}$ and $\gamma_k = 1 + \frac{2\sqrt{\hat{\alpha}_k\hat{\beta}_k}}{\hat{\alpha}_k+\hat{\beta}_k}$.

The estimation of $\hat{\alpha}_k$ and $\hat{\beta}_k$ for the dual components $\hat{h}(\hat{\lambda}_k)$ and $\hat{g}(\lambda_k)$ at the k-th iteration of primal ADMM has been described in [Xu+17b]. It is easy to verify that the model parameters $\hat{\alpha}_k$ and $\hat{\beta}_k$ of relaxed ADMM can be estimated based on the results from iteration k and an older iteration $k_0 < k$ in a similar way. If we define

$$\Delta\hat{\lambda}_k := \hat{\lambda}_k - \hat{\lambda}_{k_0} \quad \text{and} \quad \Delta\hat{h}_k := A(u_k - u_{k_0}), \tag{5.61}$$

then the parameter $\hat{\alpha}_k$ is obtained from the formula

$$\hat{\alpha}_k = \begin{cases} \hat{\alpha}_k^{\mathrm{MG}} & \text{if } \ 2\,\hat{\alpha}_k^{\mathrm{MG}} > \hat{\alpha}_k^{\mathrm{SD}} \\ \hat{\alpha}_k^{\mathrm{SD}} - \hat{\alpha}_k^{\mathrm{MG}}/2 & \text{otherwise,} \end{cases} \tag{5.62}$$

$$\hat{\alpha}_k^{\mathrm{SD}} = \frac{\langle \Delta\hat{\lambda}_k, \Delta\hat{\lambda}_k \rangle}{\langle \Delta\hat{h}_k, \Delta\hat{\lambda}_k \rangle} \ \text{ and } \ \hat{\alpha}_k^{\mathrm{MG}} = \frac{\langle \Delta\hat{h}_k, \Delta\hat{\lambda}_k \rangle}{\langle \Delta\hat{h}_k, \Delta\hat{h}_k \rangle}. \tag{5.63}$$

For a detailed derivation of these formulas, see Chapter 4.

The spectral stepsize $\hat{\beta}_k$ of $\hat{g}(\lambda_k)$ is similarly estimated with $\Delta\hat{g}_k := B(v_k - v_{k_0})$, and $\Delta\lambda_k := \lambda_k - \lambda_{k_0}$. It is important to note that $\hat{\alpha}_k$ and $\hat{\beta}_k$ are obtained from the iterates of ADMM alone, i.e., our scheme does not require the user to supply the dual problem.

Safeguarding Spectral stepsize methods for simple gradient descent are paired with a backtracking line search to guarantee convergence in case the linear model assumptions break down and an unstable stepsize is produced. ADMM methods have no analog of backtracking. Rather, we adopt the correlation criterion proposed in [Xu+17b] to test the validity of the local linear assumption, and only rely on the adaptive model when the assumptions are deemed valid. To this end, we define

$$\alpha_k^{\mathrm{cor}} = \frac{\langle \Delta\hat{h}_k, \Delta\hat{\lambda}_k \rangle}{\|\Delta\hat{h}_k\| \, \|\Delta\hat{\lambda}_k\|} \ \text{ and } \ \beta_k^{\mathrm{cor}} = \frac{\langle \Delta\hat{g}_k, \Delta\lambda_k \rangle}{\|\Delta\hat{g}_k\| \, \|\Delta\lambda_k\|}. \tag{5.64}$$

When the model assumptions (5.58) hold perfectly, the vectors $\Delta\hat{h}_k$ and $\Delta\hat{\lambda}_k$ should be highly correlated and we get $\alpha_k^{\mathrm{cor}} = 1$. When α_k^{cor} or β_k^{cor} is small, the model

assumptions are invalid and the spectral stepsize may not be effective.

The proposed method uses the following update rules

$$\tau_{k+1} = \begin{cases} \sqrt{\hat{\alpha}_k \hat{\beta}_k} & \text{if } \alpha_k^{\text{cor}} > \epsilon^{\text{cor}} \text{ and } \beta_k^{\text{cor}} > \epsilon^{\text{cor}} \\ \hat{\alpha}_k & \text{if } \alpha_k^{\text{cor}} > \epsilon^{\text{cor}} \text{ and } \beta_k^{\text{cor}} \leq \epsilon^{\text{cor}} \\ \hat{\beta}_k & \text{if } \alpha_k^{\text{cor}} \leq \epsilon^{\text{cor}} \text{ and } \beta_k^{\text{cor}} > \epsilon^{\text{cor}} \\ \tau_k & \text{otherwise,} \end{cases} \tag{5.65}$$

$$\gamma_{k+1} = \begin{cases} 1 + \frac{2\sqrt{\hat{\alpha}_k \hat{\beta}_k}}{\hat{\alpha}_k + \hat{\beta}_k} & \text{if } \alpha_k^{\text{cor}} > \epsilon^{\text{cor}} \text{ and } \beta_k^{\text{cor}} > \epsilon^{\text{cor}} \\ 1.9 & \text{if } \alpha_k^{\text{cor}} > \epsilon^{\text{cor}} \text{ and } \beta_k^{\text{cor}} \leq \epsilon^{\text{cor}} \\ 1.1 & \text{if } \alpha_k^{\text{cor}} \leq \epsilon^{\text{cor}} \text{ and } \beta_k^{\text{cor}} > \epsilon^{\text{cor}} \\ 1.5 & \text{otherwise,} \end{cases} \tag{5.66}$$

where ϵ^{cor} is a quality threshold for the curvature estimates, while $\hat{\alpha}_k$ and $\hat{\beta}_k$ are the spectral stepsizes estimated in Section 5.2.4. The update for τ_{k+1} only uses model parameters that have been accurately estimated. When the model is effective for h but not g, we use a large $\gamma_k = 1.9$ to make the v update conservative relative to the u update. When the model is effective for g but not h, we use a small $\gamma_k = 1.1$ to make the v update aggressive relative to the u update.

Applying convergence guarantee Our convergence theory requires either Assumption 5.2.1 or Assumption 5.2.2 to be satisfied, which suggests that convergence is guaranteed under "bounded adaptivity" for both penalty and relaxation parameters. These conditions can be guaranteed by explicitly adding constraints to the stepsize choice in ARADMM.

Algorithm 3 Adaptive relaxed ADMM (ARADMM)

Input: initialize v_0, λ_0, τ_0, γ_0, and $k_0 = 0$

while not converge by (5.51) **and** $k <$ maxiter **do**
  Perform relaxed ADMM, as in (5.46)–(5.49)
  if $\mathrm{mod}(k, T_f) = 1$ **then**
    $\hat{\lambda}_{k+1} = \lambda_k + \tau_k(b - Au_{k+1} - Bv_k)$
    Compute spectral stepsizes $\hat{\alpha}_k, \hat{\beta}_k$ using (5.62)
    Estimate correlations $\alpha_k^{\mathrm{cor}}, \beta_k^{\mathrm{cor}}$ using (5.64)
    Update τ_{k+1}, γ_{k+1} using (5.65) and (5.66)
    Bound τ_{k+1}, γ_{k+1} using (5.67)
    $k_0 \leftarrow k$
  else
    $\tau_{k+1} \leftarrow \tau_k$ and $\gamma_{k+1} \leftarrow \gamma_k$
  end if
  $k \leftarrow k + 1$
end while

To guarantee convergence, we simply replace the parameter updates (5.65) and (5.66) with

$$\begin{aligned} \hat{\tau}_{k+1} &= \min\left\{\tau_{k+1},\ (1 + {}^{C_{cg}}/_{k^2})\,\tau_k\right\} \\ \hat{\gamma}_{k+1} &= \min\left\{\gamma_{k+1},\ 1 + {}^{C_{cg}}/_{k^2}\right\}, \end{aligned} \tag{5.67}$$

where C_{cg} is some (large) constant. It is easily verified that the parameter sequence $(\hat{\tau}_k, \hat{\gamma}_k)$ satisfies Assumption 5.2.1. In practice, the update schemes (5.65) and (5.66) converges reliably without explicitly enforcing these conditions. We use a very large C_{cg} such that the conditions are not triggered in the first few thousand iterations and provide these constraints for theoretical interests.

ARADMM algorithm The complete *adaptive relaxed ADMM* (ARADMM) is shown in Algorithm 3. We suggest only updating the stepsize every $T_f = 2$ iterations. We suggest a fixed safeguarding threshold $\epsilon^{\mathrm{cor}} = 0.2$, which is used in all

our experiments. The overhead of the adaptive scheme is modest, requiring only a few inner product calculations.

5.2.5 Proofs of convergence theorems

We now prove that relaxed ADMM converges under Assumption 5.2.1 or 5.2.2. Let

$$y = \begin{pmatrix} u \\ \\ v \end{pmatrix} \in \mathbb{R}^{n+m}, \; z = \begin{pmatrix} u \\ \\ v \\ \\ \lambda \end{pmatrix} \in \mathbb{R}^{n+m+p}. \tag{5.68}$$

We use $y_k = (u_k, v_k)^T$ and $z_k = (u_k, v_k, \lambda_k)^T$ to denote iterates, and $y^\star = (u^\star, v^\star)^T$ and $z^\star = (u^\star, v^\star, \lambda^\star)^T$ denote optimal solutions. Set $\Delta z_k^+ = (\Delta u_k^+, \Delta v_k^+, \Delta \lambda_k^+) := z_{k+1} - z_k$, and $\Delta z_k^* = (\Delta u_k^*, \Delta v_k^*, \Delta \lambda_k^*) := z^\star - z_k$, and define

$$f(y) = h(u) + g(v), \quad F(z) = \begin{pmatrix} -A^T\lambda \\ \\ -B^T\lambda \\ \\ Au + Bv - b \end{pmatrix}. \tag{5.69}$$

Notice that $F(z)$ is monotone, which means $\forall z, z', (z - z')^T(F(z) - F(z')) \geq 0$.

Problem formulation (5.156) can be reformulated as a variational inequality (VI). The optimal solution $z^\star$ satisfies

$$\forall z, \;\; f(y) - f(y^\star) + (z - z^\star)^T F(z^\star) \geq 0. \tag{5.70}$$

Likewise, the ADMM iterates produced by steps (5.46) and (5.48) satisfy the variational inequalities

$$\begin{aligned} &\forall u,\ h(u) - h(u_{k+1}) + (u - u_{k+1})^T \\ &\qquad (\tau_k A^T(Au_{k+1} + Bv_k - b) - A^T\lambda_k) \geq 0, \end{aligned} \tag{5.71}$$

$$\begin{aligned} &\forall v,\ g(v) - g(v_{k+1}) + (v - v_{k+1})^T \\ &\qquad (\tau_k B^T(\hat{u}_{k+1} + Bv_{k+1} - b) - B^T\lambda_k) \geq 0. \end{aligned} \tag{5.72}$$

Using the definitions of y, z, $f(y)$, and $F(z)$ in (5.68, 5.69), λ in (5.49), and $\hat{u}$ in (5.47), VI (5.71) and (5.72) combine to yield

$$\begin{aligned} &f(y) - f(y_{k+1}) + (z - z_{k+1})^T \left(F(z_{k+1}) + \Omega(\Delta z_k^+, \tau_k, \gamma_k)\right) \geq 0, \\ &\Omega(\Delta z_k^+, \tau_k, \gamma_k) = \begin{pmatrix} \frac{\gamma_k - 1}{\gamma_k} A^T \Delta\lambda_k^+ - \frac{\tau_k}{\gamma_k} A^T B \Delta v_k^+ \\ 0 \\ \frac{1}{\gamma_k \tau_k} \Delta\lambda_k^+ - \frac{\gamma_k - 1}{\gamma_k} B \Delta v_k^+ \end{pmatrix}. \end{aligned} \tag{5.73}$$

We then apply VI (5.70), (5.72), and (5.73) in order to prove the following lemmas for our contraction proof, which show that the difference between iterates decreases as the iterates approach the true solution. The remaining details of the proof are in the appendix.

Lemma 5.2.1. *The iterates $z_k = (u_k, v_k, \lambda_k)^T$ generated by ADMM satisfy*

$$(B\Delta v_k^+)^T \Delta\lambda_k^+ \geq 0. \tag{5.74}$$

Lemma 5.2.2. *Let $\gamma_k \geq 1$. The optimal solution $z^\star$ and iterates z_k generated by ADMM satisfy*

$$\begin{aligned}
&\frac{2-\gamma_k}{\gamma_k}\|\tau_k B\Delta v_k^+ + \Delta\lambda_k^+\|^2 \\
&\quad \leq \gamma_k(\|\tau_k B\Delta v_k^*\|^2 + \|\Delta\lambda_k^*\|^2) \\
&\quad\quad - (2-\gamma_k)(\|\tau_k B\Delta v_{k+1}^*\|^2 + \|\Delta\lambda_{k+1}^*\|^2).
\end{aligned} \tag{5.75}$$

We are now ready to state our main convergence results. The proof of Theorem 5.2.1 is shown here in full, and leverages Lemma 5.2.2 to produce a contraction argument. The proof of Theorem 5.2.2 is extremely similar, and is shown in the appendix.

Theorem 5.2.1. *Suppose Assumption 5.2.1 holds. Then, the iterates $z_k = (u_k, v_k, \lambda_k)^T$ generated by ADMM satisfy*

$$\lim_{k\to\infty}\|r_k\| = 0 \quad \textit{and} \quad \lim_{k\to\infty}\|d_k\| = 0. \tag{5.76}$$

Proof. Assumption 5.2.1 implies

$$\frac{\gamma_k}{2-\gamma_k}\tau_k^2 \leq (1+\eta_k^2)\tau_{k-1}^2 \text{ and } \frac{\gamma_k}{2-\gamma_k} \leq (1+\eta_k^2). \tag{5.77}$$

If $\gamma_k < 2$ as in Assumption 5.2.1, then Lemma 5.2.2 shows

$$\begin{aligned}
&\frac{1}{\gamma_k}\|\tau_k B\Delta v_k^+ + \Delta\lambda_k^+\|^2 \\
\leq&\frac{\gamma_k}{2-\gamma_k}(\tau_k^2\|B\Delta v_k^*\|^2 + \|\Delta\lambda_k^*\|^2) \\
&\quad - (\tau_k^2\|B\Delta v_{k+1}^*\|^2 + \|\Delta\lambda_{k+1}^*\|^2) \qquad (5.78)\\
\leq&(1+\eta_k^2)(\tau_{k-1}^2\|B\Delta v_k^*\|^2 + \|\Delta\lambda_k^*\|^2) \\
&\quad - (\tau_k^2\|B\Delta v_{k+1}^*\|^2 + \|\Delta\lambda_{k+1}^*\|^2), \qquad (5.79)
\end{aligned}$$

where (5.77) is used to get from (5.78) to (5.79). Accumulating inequality (5.79) from $k = 0$ to N shows

$$\begin{aligned}
&\sum_{k=0}^{N}\prod_{t=k+1}^{N}(1+\eta_t^2)\frac{1}{\gamma_k}\|\tau_k B\Delta v_k^+ + \Delta\lambda_k^+\|^2 \\
&\quad\leq \prod_{k=1}^{N}(1+\eta_t^2)(\tau_0^2\|B\Delta v_0^*\|^2 + \|\Delta\lambda_0^*\|^2). \qquad (5.80)
\end{aligned}$$

Assumption 5.2.1 also implies $\prod_{t=1}^{\infty}(1+\eta_t^2) < \infty$, and $\prod_{t=k+1}^{N}(1+\eta_t^2)\frac{1}{\gamma_k} \geq \frac{1}{\gamma_k} > 1/2$. Then, (5.80) indicates $\sum_{k=0}^{\infty}\|\tau_k B\Delta v_k^+ + \Delta\lambda_k^+\|^2 < \infty$, and

$$\lim_{k\to\infty}\|\tau_k B\Delta v_k^+ + \Delta\lambda_k^+\|^2 = 0. \qquad (5.81)$$

Now, from Lemma 5.2.1, $(B\Delta v_k^+)^T \Delta\lambda_k^+ \geq 0$, and so

$$\lim_{k\to\infty} \|\Delta\lambda_k^+\|^2 \leq \lim_{k\to\infty} \|\tau_k B\Delta v_k^+ + \Delta\lambda_k^+\|^2 = 0, \tag{5.82}$$

$$\lim_{k\to\infty} \|\tau_k B\Delta v_k^+\|^2 \leq \lim_{k\to\infty} \|\tau_k B\Delta v_k^+ + \Delta\lambda_k^+\|^2 = 0. \tag{5.83}$$

The residuals r_k, d_k in (5.50) satisfy

$$r_k = \frac{1}{\gamma_k\tau_k}\Delta\lambda_{k-1}^+ - \frac{\gamma_k - 1}{\gamma_k} B\Delta v_{k-1}^+, \tag{5.84}$$

$$d_k = \tau_k A^T B\Delta v_{k-1}^+, \tag{5.85}$$

from which we get

$$\lim_{k\to\infty} \|r_k\| \leq \lim_{k\to\infty} \frac{1}{\gamma_k\tau_k}\|\Delta\lambda_{k-1}^+\| + \frac{\gamma_k - 1}{\gamma_k\tau_{k-1}^2}\|\tau_{k-1}B\Delta v_{k-1}^+\| = 0, \text{ and} \tag{5.86}$$

$$\begin{aligned}\lim_{k\to\infty} \|d_k\| &\leq \lim_{k\to\infty} \|A\|\|\tau_k B\Delta v_{k-1}^+\| \\ &\leq \lim_{k\to\infty} \sqrt{1+\eta_k^2}\|A\|\,\|\tau_{k-1}B\Delta v_{k-1}^+\| = 0.\end{aligned}$$

□

Similar methods can be used to prove the following about convergence under Assumption 5.2.2. The proof of the following theorem is given in the appendix.

Theorem 5.2.2. *Suppose Assumption 5.2.2 holds. Then, the iterates* $z_k =$

$(u_k, v_k, \lambda_k)^T$ *generated by ADMM satisfy*

$$\lim_{k\to\infty} \|r_k\| = 0 \quad \textit{and} \quad \lim_{k\to\infty} \|d_k\| = 0. \tag{5.87}$$

5.2.6 Appendix:proofs of lemmas and theorems

Proof of Lemma 5.2.1

Proof. Using the dual updates (5.49), VI (5.72) can be rewritten as

$$\forall v,\ g(v) - g(v_{k+1}) - (Bv - Bv_{k+1})^T \lambda_{k+1} \geq 0. \tag{5.88}$$

Similarly, in the previous iteration,

$$\forall v,\ g(v) - g(v_k) - (Bv - Bv_k)^T \lambda_k \geq 0. \tag{5.89}$$

After letting $v = v_k$ in (5.88) and $v = v_{k+1}$ in (5.89), we sum the two inequalities together to conclude

$$(Bv_{k+1} - Bv_k)^T (\lambda_{k+1} - \lambda_k) \geq 0. \tag{5.90}$$

□

Lemma 5.2.3. *The optimal solution* $z^\star = (u^\star, v^\star, \lambda^\star)^T$ *and sequence* $z_k =$

$(u_k, v_k, \lambda_k)^T$ *generated by ADMM satisfy*

$$\begin{aligned}
&(\tau_k B\Delta v_{k+1}^* + \Delta\lambda_{k+1}^*)^T(\tau_k B\Delta v_k^+ + \Delta\lambda_k^+) \\
&\geq \frac{1-\gamma_k}{\gamma_k}\|\tau_k B\Delta v_k^+ + \Delta\lambda_k^+\|^2 \\
&+ \gamma_k((\tau_k B\Delta v_k^*)^T\Delta\lambda_k^* - (\tau_k B\Delta v_{k+1}^*)^T\Delta\lambda_{k+1}^*).
\end{aligned} \tag{5.91}$$

Proof of Lemma 5.2.3

Proof. We replace $y = y^\star, z = z^\star$ in VI (5.73) and $y = y_{k+1}, z = z_{k+1}$ in VI (5.70), and sum the two inequalities to get

$$(\Delta z_{k+1}^*)^T\Omega(\Delta z_k^+, \tau_k, \gamma_k) \quad \geq \quad (\Delta z_{k+1}^*)^T(F(z^\star) - F(z_{k+1})). \tag{5.92}$$

From (5.73), the monotonicity of $F(z)$, and $\Omega(\Delta z_k^+, \tau_k, \gamma_k)$, we have

$$\begin{aligned}
&(\tau_k A\Delta u_{k+1}^*)^T((\gamma_k - 1)\Delta\lambda_k^+ - \tau_k B\Delta v_k^+) \\
&+ (\Delta\lambda_{k+1}^*)^T(\Delta\lambda_k^+ + (1-\gamma_k)\tau_k B\Delta v_k^+) \geq 0.
\end{aligned} \tag{5.93}$$

Using the feasibility of optimal solution $Au^\star + Bv^\star = b$, λ_{k+1} in (5.49) and $\hat{u}_{k+1}$ in (5.47), we have

$$\tau_k A\Delta u_{k+1}^* = \frac{1}{\gamma_k}\Delta\lambda_k^+ + \frac{1-\gamma_k}{\gamma_k}\tau_k B\Delta v_k^+ - \tau_k B\Delta v_{k+1}^*. \tag{5.94}$$

We now substitute (5.94) into (5.93) and simplify to get,

$$\begin{aligned}(\tau_k B\Delta v_{k+1}^* + \Delta\lambda_{k+1}^*)^T(\tau_k B\Delta v_k^+ + \Delta\lambda_k^+) \geq \\ \frac{1-\gamma_k}{\gamma_k}\|\tau_k B\Delta v_k^+ + \Delta\lambda_k^+\|^2 + \gamma_k(\tau_k B\Delta v_k^+)^T\Delta\lambda_k^+ \\ + \gamma_k((\tau_k B\Delta v_{k+1}^*)^T\Delta\lambda_k^+ + (\tau_k B\Delta v^+)^T\Delta\lambda_{k+1}^*). \end{aligned} \tag{5.95}$$

We can use the fact that

$$\Delta\lambda_k^* = \Delta\lambda_{k+1}^* + \Delta\lambda_k^+ \text{ and } \Delta v_k^* = \Delta v_{k+1}^* + \Delta v_k^+, \tag{5.96}$$

to get

$$\begin{aligned}(\tau_k B\Delta v_{k+1}^*)^T\Delta\lambda_k^+ + (\tau_k B\Delta v^+)^T\Delta\lambda_{k+1}^* \\ = (\tau_k B\Delta v_k^*)^T\Delta\lambda_k^* - (\tau_k B\Delta v_{k+1}^*)^T\Delta\lambda_{k+1}^* \\ - (\tau_k B\Delta v_k^+)^T\Delta\lambda_k^+. \end{aligned} \tag{5.97}$$

Finally, we substitute (5.97) into (5.95) to get (5.91). □

Proof of Lemma 5.2.2

Proof. Begin by deriving

$$\begin{aligned}
&\|\tau_k B\Delta v_k^* + \Delta\lambda_k^*\|^2 \\
&\quad = \|(\tau_k B\Delta v_{k+1}^* + \Delta\lambda_{k+1}^*) + (\tau_k B\Delta v_k^+ + \Delta\lambda_k^+)\|^2 && (5.98)\\
&\quad = \|\tau_k B\Delta v_{k+1}^* + \Delta\lambda_{k+1}^*\|^2 + \|\tau_k B\Delta v_k^+ + \Delta\lambda_k^+\|^2 \\
&\qquad + 2(\tau_k B\Delta v_{k+1}^* + \Delta\lambda_{k+1}^*)^T(\tau_k B\Delta v_k^+ + \Delta\lambda_k^+) && (5.99)\\
&\quad \geq \|\tau_k B\Delta v_{k+1}^* + \Delta\lambda_{k+1}^*\|^2 + \frac{2-\gamma_k}{\gamma_k}\|\tau_k B\Delta v_k^+ + \Delta\lambda_k^+\|^2 \\
&\quad + 2\gamma_k((\tau_k B\Delta v_k^*)^T\Delta\lambda_k^* - (\tau_k B\Delta v_{k+1}^*)^T\Delta\lambda_{k+1}^*), && (5.100)
\end{aligned}$$

where (5.96) is used for (5.98), and Lemma 5.2.3 is used for (5.100). We now have

$$
\begin{aligned}
&\frac{2-\gamma_k}{\gamma_k}\|\tau_k B\Delta v_k^+ + \Delta\lambda_k^+\|^2 \\
&\quad \leq \|\tau_k B\Delta v_k^* + \Delta\lambda_k^*\|^2 - \|\tau_k B\Delta v_{k+1}^* + \Delta\lambda_{k+1}^*\|^2 \\
&\quad -2\gamma_k((\tau_k B\Delta v_k^*)^T\Delta\lambda_k^* - (\tau_k B\Delta v_{k+1}^*)^T\Delta\lambda_{k+1}^*) \qquad (5.101)\\
&\quad = \|\tau_k B\Delta v_k^*\|^2 + \|\Delta\lambda_k^*\|^2 - \|\tau_k B\Delta v_{k+1}^*\|^2 - \|\Delta\lambda_{k+1}^*\|^2 \\
&\qquad - 2(\gamma_k-1)(\tau_k B\Delta v_k^*)^T\Delta\lambda_k^* \\
&\qquad - 2(\gamma_k-1)(-\tau_k B\Delta v_{k+1}^*)^T\Delta\lambda_{k+1}^* \qquad (5.102)\\
&\quad = \gamma_k(\|\tau_k B\Delta v_k^*\|^2 + \|\Delta\lambda_k^*\|^2) \\
&\qquad - (2-\gamma_k)(\|\tau_k B\Delta v_{k+1}^*\|^2 + \|\Delta\lambda_{k+1}^*\|^2) \\
&\qquad - (\gamma_k-1)\|\tau_k B\Delta v_k^* + \Delta\lambda_k^*\|^2 \\
&\qquad - (\gamma_k-1)\|\tau_k B\Delta v_{k+1}^* - \Delta\lambda_{k+1}^*\|^2 \qquad (5.103)\\
&\quad \leq \gamma_k(\|\tau_k B\Delta v_k^*\|^2 + \|\Delta\lambda_k^*\|^2) \\
&\qquad - (2-\gamma_k)(\|\tau_k B\Delta v_{k+1}^*\|^2 + \|\Delta\lambda_{k+1}^*\|^2). \qquad (5.104)
\end{aligned}
$$

□

Proof of Theorem 5.2.2

Proof. When $\gamma_k < 2$ as in Assumption 5.2.2, Lemma 5.2.2 suggests

$$\begin{aligned}
&\frac{1}{\gamma_k}\|B\Delta v_k^+ + \frac{1}{\tau_k}\Delta\lambda_k^+\|^2 \\
\leq&\frac{\gamma_k}{2-\gamma_k}(\|B\Delta v_k^*\|^2 + \frac{1}{\tau_k^2}\|\Delta\lambda_k^*\|^2) \\
&- (\|B\Delta v_{k+1}^*\|^2 + \frac{1}{\tau_k^2}\|\Delta\lambda_{k+1}^*\|^2).
\end{aligned} \tag{5.105}$$

Assumption 5.2.2 also suggests

$$\frac{\gamma_k}{(2-\gamma_k)\tau_k^2} \leq \frac{1+\theta_k^2}{\tau_{k-1}^2} \text{ and } \frac{\gamma_k}{2-\gamma_k} \leq (1+\theta_k^2). \tag{5.106}$$

Then (5.105) leads to

$$\begin{aligned}
&\frac{1}{\gamma_k}\|\frac{1}{\tau_k}\Delta v_k^+ + B\Delta\lambda_k^+\|^2 \\
\leq&(1+\theta_k^2)(\|B\Delta v_k^*\|^2 + \frac{1}{\tau_{k-1}^2}\|\Delta\lambda_k^*\|^2) \\
&- (\|B\Delta v_{k+1}^*\|^2 + \frac{1}{\tau_k^2}\|\Delta\lambda_{k+1}^*\|^2).
\end{aligned} \tag{5.107}$$

Accumulating (5.107) from $k = 0$ to get

$$\begin{aligned}
&\sum_{k=0}^{N}\prod_{t=k+1}^{N}(1+\theta_t^2)\frac{1}{\gamma_k}\|B\Delta v_k^+ + \frac{1}{\tau_k}\Delta\lambda_k^+\|^2 \\
&\leq \prod_{k=1}^{N}(1+\theta_t^2)(\|B\Delta v_0^*\|^2 + \frac{1}{\tau_0^2}\|\Delta\lambda_0^*\|^2).
\end{aligned} \tag{5.108}$$

Assumption 5.2.2 suggests $\prod_{t=1}^{\infty}(1+\theta_t^2) < \infty$, and $\prod_{t=k+1}^{N}(1+\theta_t^2)\frac{1}{\gamma_k} \geq \frac{1}{\gamma_k} > 0.5$.

Then (5.108) indicates $\sum_{k=0}^{\infty} \|B\Delta v_k^+ + \frac{1}{\tau_k}\Delta\lambda_k^+\|^2 < \infty$. Hence

$$\lim_{k\to\infty} \|B\Delta v_k^+ + \frac{1}{\tau_k}\Delta\lambda_k^+\|^2 = 0. \tag{5.109}$$

Since $(B\Delta v_k^+)^T\Delta\lambda_k^+ \geq 0$ as in Lemma 5.2.1,

$$\lim_{k\to\infty} \|\frac{1}{\tau_k}\Delta\lambda_k^+\|^2 \leq \lim_{k\to\infty} \|B\Delta v_k^+ + \frac{1}{\tau_k}\Delta\lambda_k^+\|^2 = 0 \tag{5.110}$$

$$\lim_{k\to\infty} \|B\Delta v_k^+\|^2 \leq \lim_{k\to\infty} \|B\Delta v_k^+ + \frac{1}{\tau_k}\Delta\lambda_k^+\|^2 = 0. \tag{5.111}$$

The residuals r_k, d_k in (5.50) then satisfy

$$r_k = \frac{1}{\gamma_k\tau_k}\Delta\lambda_{k-1}^+ - \frac{\gamma_k - 1}{\gamma_k}B\Delta v_{k-1}^+ \tag{5.112}$$

$$d_k = \tau_k A^T B\Delta v_{k-1}^+. \tag{5.113}$$

We finally have

$$\begin{aligned}\lim_{k\to\infty} \|r_k\| &\leq \lim_{k\to\infty} \frac{1}{\gamma_k}\|\frac{1}{\tau_k}\Delta\lambda_{k-1}^+\| + \frac{\gamma_k - 1}{\gamma_k}\|B\Delta v_{k-1}^+\| \\ &\leq \lim_{k\to\infty} \frac{\sqrt{1+\theta_k^2}}{\gamma_{k-1}}\|\frac{1}{\tau_k}\Delta\lambda_{k-1}^+\| \\ &\quad + \frac{\gamma_k - 1}{\gamma_k}\|B\Delta v_{k-1}^+\| = 0, \text{ and}\end{aligned} \tag{5.114}$$

$$\begin{aligned}\lim_{k\to\infty} \|d_k\| &\leq \lim_{k\to\infty} |A|\|\tau_k B\Delta v_{k-1}^+\| \\ &\leq \lim_{k\to\infty} (1+\eta_k^2)\tau_k|A|\,\|B\Delta v_{k-1}^+\| = 0.\end{aligned} \tag{5.115}$$

□

Equivalence of relaxed ADMM and relaxed DRS in Section 5.2.3

Proof. Referring back to the ADMM steps (5.46)–(5.49), and defining $\hat{\lambda}_{k+1} = \lambda_k + \tau_k(b - Au_{k+1} - Bv_k)$, the optimality condition for the minimization of (5.46) is

$$0 \in \partial h(u_{k+1}) - A^T\lambda_k - \tau_k A^T(b - Au_{k+1} - Bv_k) \tag{5.116}$$

$$= \partial h(u_{k+1}) - A^T\hat{\lambda}_{k+1}, \tag{5.117}$$

which is equivalent to $A^T\hat{\lambda}_{k+1} \in \partial h(u_{k+1})$, thus[1] $u_{k+1} \in \partial h^*(A^T\hat{\lambda}_{k+1})$. A similar argument using the optimality condition for (5.48) leads to $v_{k+1} \in \partial g^*(B^T\lambda_{k+1})$. Recalling (5.54), we arrive at

$$Au_{k+1} - b \in \partial\hat{h}(\hat{\lambda}_{k+1}) \quad \text{and} \quad Bv_{k+1} \in \partial\hat{g}(\lambda_{k+1}). \tag{5.118}$$

[1] An important property relating f and f^* is that $y \in \partial f(x)$ if and only if $x \in \partial f^*(y)$ [Roc70].

Using these identities, we finally have

$$\hat{\lambda}_{k+1} = \lambda_k + \tau_k(b - Au_{k+1} - Bv_k) \tag{5.119}$$

$$\in \lambda_k - \tau_k\big(\partial\hat{h}(\hat{\lambda}_{k+1}) + \partial\hat{g}(\lambda_k)\big) \tag{5.120}$$

$$\lambda_{k+1} = \lambda_k + \tau_k(b - \hat{u}_{k+1} - Bv_{k+1}) \tag{5.121}$$

$$= \lambda_k + \gamma_k\tau_k(b - Au_{k+1} - Bv_{k+1})$$

$$+ (1-\gamma_k)\tau_k(Bv_k - Bv_{k+1}) \tag{5.122}$$

$$\in \lambda_k - \tau_k\big(\partial\hat{h}(\hat{\lambda}_{k+1}) + \partial\hat{g}(\lambda_{k+1})\big)$$

$$+ (1-\gamma_k)\tau_k\left(\partial\hat{g}(\lambda_k) - \partial\hat{g}(\lambda_{k+1})\right), \tag{5.123}$$

showing that the sequences $(\lambda_k)_{k\in\mathbb{N}}$ and $(\hat{\lambda}_k)_{k\in\mathbb{N}}$ satisfy the same conditions (5.55) and (5.56) as $(\zeta_k)_{k\in\mathbb{N}}$ and $(\hat{\zeta}_k)_{k\in\mathbb{N}}$, thus proving that ADMM for problem (5.156) is equivalent to DRS for its dual (5.54). □

Proof of Proposition 5.2.1 in Section 5.2.4

Proof. Rearrange DRS step (5.56) to get

$$0 \in \frac{\zeta_{k+1} - \zeta_k}{(1-\gamma)\tau} + \frac{\gamma}{1-\gamma}\,\partial\hat{h}(\hat{\zeta}_{k+1}) - \partial\hat{g}(\zeta_k) + \frac{1}{1-\gamma}\partial\hat{g}(\zeta_{k+1}). \tag{5.124}$$

Combine DRS step (5.55) and (5.124) to get

$$0 \in \frac{1}{\tau}\left(\frac{\zeta_{k+1}}{1-\gamma} + \hat{\zeta}_{k+1} - \frac{2-\gamma}{1-\gamma}\zeta_k\right) + \frac{1}{1-\gamma}\left(\partial\hat{h}(\hat{\zeta}_{k+1}) + \hat{g}(\zeta_{k+1})\right). \tag{5.125}$$

Inserting the linear assumption (5.57) to DRS step (5.55), we can explicitly get the update for $\hat{\zeta}_{k+1}$ as

$$\hat{\zeta}_{k+1} = \frac{1-\beta\,\tau}{1+\alpha\,\tau}\zeta_k - \frac{a\tau+b\tau}{1+\alpha\,\tau}, \tag{5.126}$$

where $a \in \Psi$ and $b \in \Phi$. Inserting the linear model (5.57) into (5.125), we get

$$\zeta_{k+1} = \frac{\gamma-1-\alpha\tau}{1+\beta\tau}\hat{\zeta}_{k+1} + \frac{2-\gamma}{1+\beta\tau}\zeta_k - \frac{(a+b)\tau}{1+\beta\tau} \tag{5.127}$$

$$= \zeta_k - \gamma\tau\,\frac{(\alpha+\beta)\zeta_k + (a+b)}{(1+\alpha\tau)(1+\beta\tau)}, \tag{5.128}$$

where the second equality results from using the expression for $\hat{\zeta}_{k+1}$ from (5.126).

The residual r_{DR} at ζ_{k+1} is simply the magnitude of the subgradient (corresponding to elements $a \in \Psi$ and $b \in \Phi$) of the objective and is given by

$$r_{DR} = \|(\alpha+\beta)\zeta_{k+1} + (a+b)\| \tag{5.129}$$

$$= \left|1 - \frac{\gamma\tau(\alpha+\beta)}{(1+\alpha\,\tau)(1+\beta\,\tau)}\right| \cdot \|(\alpha+\beta)\zeta_k + (a+b)\|, \tag{5.130}$$

where ζ_{k+1} in (5.130) was substituted with (5.128). The optimal parameters minimize the residual

$$\tau, \gamma = \arg\min_{\tau,\gamma} r_{\mathrm{DR}}$$

$$= \arg\min_{\tau,\gamma} \left|1 - \frac{\gamma\tau(\alpha+\beta)}{(1+\alpha\,\tau)(1+\beta\,\tau)}\right|. \tag{5.131}$$

This residual has optimal value of zero when

$$\gamma_k = 1 + \frac{1 + \alpha\beta\tau_k^2}{(\alpha + \beta)\tau_k}.$$

□

5.2.7 Experiments

The proposed ARADMM is implemented as shown in Algorithm 3. We also implemented vanilla ADMM, (non-adaptive) relaxed ADMM, ADMM with residual balancing (RB), and adaptive ADMM (AADMM) for comparison.

The relaxation parameter for the non-adaptive relaxed ADMM is fixed at $\gamma_k = 1.5$ as suggested in [EB92]. The parameters of RB and AADMM are selected as in [He+00; Boy+11; Xu+17b]. The initial penalty $\tau_0 = 1/10$ and initial relaxation $\gamma_0 = 1$ are used for all problems except the canonical QP problem, where initial parameters are set to the geometric mean of the maximum and minimum eigenvalues of matrix Q, as proposed for quadratic problems in [RDC14].

For each problem, the same randomly generated initial variables v_0, λ_0 are used for ADMM and its variant methods. As suggested by [He+00; Xu+17b], the adaptivity of RB and AADMM is stopped after 1000 iterations to guarantee convergence.

Convergence results Table 5.3 reports the convergence speed of ADMM and its variants for various applications. More experimental results including the table of

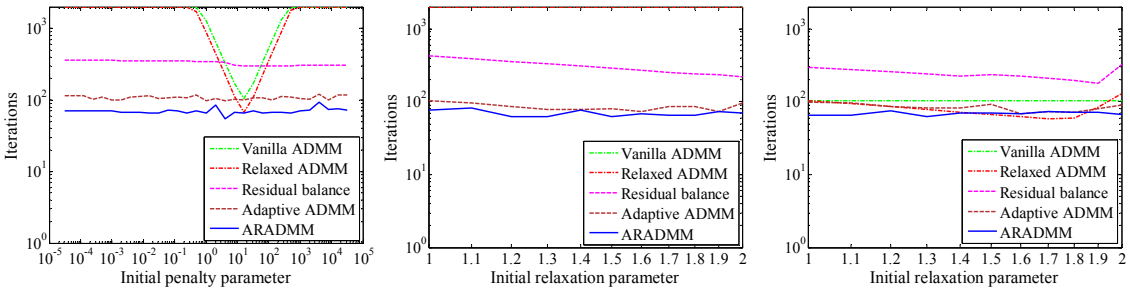


Figure 5.7: Sensitivity of convergence speed for the synthetic problem of EN regularized linear regression. (left) sensitivity to the initial penalty τ_0; (middle) sensitivity to relaxation γ_0; (right) sensitivity to relaxation γ_0 when optimal τ_0 is selected by grid search.

more test cases, the convergence curves, and visual results of image restoration and robust PCA for face decomposition are provided in the supplementary material. Relaxed ADMM often outperforms vanilla ADMM, but does not compete with adaptive methods like RB, AADMM and ARADMM. The proposed ARADMM performs best in all the test cases.

Sensitivity to initialization We study the sensitivity of the different ADMM variants to the initial penalty (τ_0) and initial relaxation parameter (γ_0). Fig. 5.7 presents iteration counts for a wide range of values of τ_0, γ_0, for elastic net regression with synthetic datasets. In the left and center plots we fix one of τ_0, γ_0 and vary the other. The number of iterations needed to convergence is plotted as the algorithm parameters vary. In the right plot, we use a grid search to find the *optimal* τ_0 for different values of γ_0. Fig. 5.7 (left) shows that adaptive methods are relatively stable with respect to the initial penalty τ_0, while ARADMM outperforms RB and AADMM in all choices of initial τ_0. Fig. 5.7 (middle) suggests that the relaxation γ_0 is generally less important than τ_0. When a bad value of τ is chosen, it is unlikely that a good choice of γ can compensate. The proposed ARADMM that jointly adjusts τ, γ is

generally better than simply adding the relaxation to the existing adaptive methods RB and AADMM.

Fig. 5.7 (right) shows the sensitivity to γ when using a grid search to choose the optimal τ_0. This optimal τ_0 significantly improves the performance of vanilla ADMM and relaxed ADMM (which use the same τ_0 for all iterations). Even when using the optimal stepsize for the non-adaptive methods, ARADMM is superior to or competitive with the non-adaptive methods. Note that this experiment is meant to show a best-case scenario for the non-adaptive methods; in practice the user generally has no knowledge of the optimal value of τ. Adaptive methods achieve optimal or near-optimal performance without an expensive grid search.

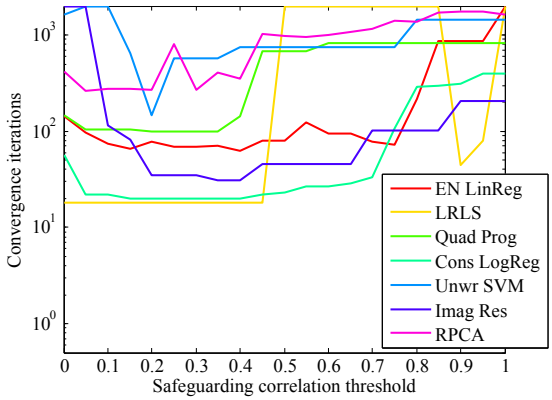


Figure 5.8: Sensitivity of convergence speed to safeguarding threshold ϵ^{cor} for proposed ARADMM. Synthetic problems ('cameraman' for TVIR, and 'FaceSet1' for RPCA) of various applications are studied. Best viewed in color.

Sensitivity to safeguarding Finally, Fig. 5.8 presents iteration counts when applying ARADMM with various safeguarding correlation thresholds ϵ^{cor}. When $\epsilon^{\text{cor}} = 0$, the calculated adaptive parameters based on curvature estimations are always accepted, and when $\epsilon^{\text{cor}} = 1$ the parameters are never changed. The proposed AADMM

method is insensitive to ϵ^{cor} and performs well for a wide range of $\epsilon^{\text{cor}} \in [0.1, 0.4]$ for various applications, except for unwrapping SVM and RPCA. Though tuning such "hyper-parameters" may improve the performance of ARADMM for some applications, the fixed $\epsilon^{\text{cor}} = 0.2$ performs well in all our experiments (seven applications and over fifty test cases, a full list is in the supplementary material). The proposed ARADMM is fully automated and performs well without parameter tuning.

5.2.8 Summarization

We have proposed an adaptive method for jointly tuning the penalty and relaxation parameters of relaxed ADMM without user oversight. We have analyzed adaptive relaxed ADMM schemes, and provided conditions for which convergence is guaranteed. Experiments on a wide range of machine learning, computer vision, and image processing benchmarks have demonstrated that the proposed adaptive method (often significantly) outperforms other ADMM variants without user oversight or parameter tuning. The new adaptive method improves the applicability of relaxed ADMM by facilitating fully automated solvers that exhibit fast convergence and are usable by non-expert users.

Application	Dataset	Vanilla ADMM	Relaxed ADMM	Residual balance	Adaptive ADMM	**Proposed ARADMM**
Elastic net regression	Synthetic	2000+(.642)	2000+(.660)	424(.144)	102(.051)	**70(.026)**
	MNIST	1225(29.4)	816(19.9)	94(2.28)	41(.943)	**21(.549)**
	CIFAR10	2000+(690)	2000+(697)	556(193)	2000+(669)	**94(31.7)**
	News20	2000+(1.21e4)	2000+(9.16e3)	227(914)	104(391)	**71(287)**
	Rcv1	2000+(1.20e3)	1823(802)	196(79.1)	104(35.7)	**64(26.0)**
	Realsim	2000+(4.26e3)	2000+(4.33e3)	341(355)	152(125)	**107(88.2)**
Low rank least squares	Synthetic	2000+(118)	2000+(116)	268(15.1)	26(1.55)	**18(1.04)**
	German	2000+(4.72)	2000+(4.72)	642(1.52)	130(.334)	**52(.125)**
	Spectf	2000+(2.70)	2000+(2.74)	336(.455)	162(.236)	**105(.150)**
	MNIST	200+(1.86e3)	200+(2.08e3)	200+(3.29e3)	200+(3.46e3)	**38(658)**
	CIFAR10	200+(7.24e3)	200+(1.33e4)	53(1.60e3)	8(208)	**6(156)**
QP and dual SVM	Synthetic	1224(11.5)	823(7.49)	626(5.93)	170(1.57)	**100(.914)**
	German	2000+(58.8)	2000+(61.8)	1592(45.0)	1393(38.9)	**1238(34.9)**
	Spectf	2000+(.846)	2000+(.777)	169(.070)	175(.086)	**53(.026)**
Consensus logistic regression	Synthetic	590(9.93)	391(6.97)	70(1.23)	35(.609)	**20(.355)**
	German	2000+(34.3)	2000+(66.6)	151(2.60)	35(.691)	**26(.580)**
	Spectf	1005(20.1)	667(14.4)	117(1.98)	145(1.63)	**85(1.07)**
	MNIST	200+(2.99e3)	200+(3.47e3)	200+(1.37e3)	49(536)	**28(333)**
	CIFAR10	200+(593)	200+(2.08e3)	200+(1.54e3)	131(165)	**19(33.7)**
Unwrapping SVM	Synthetic	2000+(1.13)	1418(.844)	2000+(1.16)	355(.229)	**147(.094)**
	German	753(1.88)	560(1.37)	2000+(4.98)	572(1.44)	**213(.545)**
	Spectf	567(.203)	367(.112)	567(.185)	207(.068)	**149(.052)**
	MNIST	128(130)	118(111)	163(153)	200+(217)	**67(71.0)**
	CIFAR10	200+(512)	200+(532)	200+(516)	89(285)	**57(143)**
Image denoising	Barbara	262(35.0)	175(23.6)	74(10.0)	59(8.67)	**38(5.57)**
	Cameraman	311(8.96)	208(5.89)	82(2.29)	88(2.76)	**35(1.08)**
	Lena	347(46.3)	232(31.3)	94(12.5)	68(9.70)	**39(5.58)**
Robust PCA	FaceSet1	2000+(41.1)	1507(30.3)	560(11.1)	561(11.9)	**267(5.65)**
	FaceSet2	2000+(41.1)	2000+(41.4)	263(5.54)	388(9.00)	**188(4.02)**
	FaceSet3	2000+(39.4)	1843(36.3)	375(7.44)	473(9.89)	**299(6.27)**

Table 5.3: Iterations (and runtime in seconds) for various applications. Absence of convergence after n iterations is indicated as $n+$.

5.3 Adaptive Consensus ADMM

5.3.1 Introduction

Consensus ADMM [Boy+11] solves minimization problems involving a composite objective $f(v) = \sum_i f_i(v)$, where worker i stores the data needed to compute f_i, and so is well suited for distributed model fitting problems [Boy+11; ZK14; Son+16; Cha+16a; Gol+16; Tay+16]. To distribute this problem, consensus methods assign a separate copy of the unknowns, u_i, to each worker, and then apply ADMM to solve

$$\min_{u_i \in \mathbb{R}^d, v \in \mathbb{R}^d} \sum_{i=1}^{N} f_i(u_i) + g(v), \quad \text{subject to} \quad u_i = v, \tag{5.132}$$

where v is the "central" copy of the unknowns, and $g(v)$ is a regularizer. The consensus problem (5.132) coincides with (5.156) by defining $u = (u_1; \ldots; u_N) \in \mathbb{R}^{dN}$, $A = I_{dN} \in \mathbb{R}^{dN \times dN}$, and $B = -(I_d; \ldots; I_d) \in \mathbb{R}^{dN \times d}$, where I_d represents the $d \times d$ identity matrix.

We propose an adaptive consensus ADMM (ACADMM) method to automate local algorithm parameters selection. Instead of estimating one global penalty parameter for all workers, different local penalty parameters are estimated using the local curvature of subproblems on each node.

In the following, we use the subscript i to denote iterates computed on the ith node, superscript k is the iteration number, $\lambda_{i,k}$ is the dual vector of Lagrange mul-

tipliers, and $\{\tau_{i,k}\}$ are iteration/worker-specific penalty parameters (contrasted with the single constant penalty parameter τ of "vanilla" ADMM). Consensus methods apply ADMM to (5.132), resulting in the steps

$$u_{i,k+1} = \arg\min_{u_i} f_i(u_i) + \frac{\tau_{i,k}}{2}\|v_k - u_i + \frac{\lambda_{i,k}}{\tau_{i,k}}\|^2 \tag{5.133}$$

$$v_{k+1} = \arg\min_{v} g(v) + \sum_{i=1}^{N} \frac{\tau_{i,k}}{2}\|v - u_{i,k+1} + \frac{\lambda_{i,k}}{\tau_{i,k}}\|^2 \tag{5.134}$$

$$\lambda_{i,k+1} = \lambda_{i,k} + \tau_{i,k}(v_{k+1} - u_{i,k+1}). \tag{5.135}$$

The primal and dual residuals, r^k and d^k, are used to monitor convergence.

$$r_k = \begin{pmatrix} r_{1,k} \\ \vdots \\ r_{N,k} \end{pmatrix}, \; d_k = \begin{pmatrix} d_{1,k} \\ \vdots \\ d_{N,k} \end{pmatrix}, \quad \begin{cases} r_{i,k} = v_k - u_{i,k} \\ \\ d_{i,k} = \tau_{i,k}(v_{k-1} - v_k). \end{cases} \tag{5.136}$$

The primal residual r^k approaches zero when the iterates accurately satisfy the linear constraints in (5.132), and the dual residual d^k approaches zero as the iterates near a minimizer of the objective. Iteration can be terminated when

$$\begin{aligned} &\|r_k\|^2 \le \epsilon^{tol} \max\{\sum_{i=1}^{N} \|u_{i,k}\|^2, N\|v_k\|^2\} \\ &\text{and} \quad \|d_k\|^2 \le \epsilon^{tol} \sum_{i=1}^{N} \|\lambda_{i,k}\|^2, \end{aligned} \tag{5.137}$$

where ϵ^{tol} is the stopping tolerance. The residuals in (5.136) and stopping criterion in (5.137) are adopted from the general problem [Boy+11] to the consensus problem. The observation that residuals r_k, d_k can be decomposed into "local residuals"

$r_{i,k}, d_{i,k}$ has been exploited to generalize the residual balancing method [He+00] for distributed consensus problems [Son+16].

To address the issue of how to automatically tune parameters on each node for optimal performance, we propose *adaptive consensus ADMM* (ACADMM), which sets worker-specific penalty parameters by exploiting curvature information. We derive our method from the dual interpretation of ADMM – *Douglas-Rachford splitting* (DRS) – using a diagonal penalty matrix. We then derive the spectral stepsizes for consensus problems by assuming the curvatures of the objectives are diagonal matrices with diverse parameters on different nodes. At last, we discuss the practical computation of the spectral stepsizes from consensus ADMM iterates and apply our theory in Chapter 3 to guarantee convergence.

5.3.2 Dual interpretation of generalized ADMM

The dual form of problem (5.156) can be written

$$\min_{\lambda \in \mathbb{R}^p} \underbrace{f^*(A^T\lambda) - \langle \lambda, b\rangle}_{\hat{f}(\lambda)} + \underbrace{g^*(B^T\lambda)}_{\hat{g}(\lambda)}, \tag{5.138}$$

where λ denotes the dual variable, while f^*, g^* denote the Fenchel conjugate of f, g [Roc70]. It is known that ADMM steps for the primal problem (5.156) are equivalent to performing *Douglas-Rachford splitting* (DRS) on the dual problem (5.138) [EB92; Xu+17b]. In particular, the generalized ADMM iterates satisfy the

DRS update formulas

$$0 \in (T_k)^{-1}(\hat{\lambda}_{k+1} - \lambda_k) + \partial \hat{f}(\hat{\lambda}_{k+1}) + \partial \hat{g}(\lambda_k) \tag{5.139}$$

$$0 \in (T_k)^{-1}(\lambda_{k+1} - \lambda_k) + \partial \hat{f}(\hat{\lambda}_{k+1}) + \partial \hat{g}(\lambda_{k+1}), \tag{5.140}$$

where $\hat{\lambda}_{i,k+1} = \lambda_{i,k} + \tau_{i,k}(v_{k+1} - u_{i,k})$ denotes the intermediate variable. We now prove the equivalence of generalized ADMM and DRS.

Proof. The optimality condition for ADMM step (3.1) is

$$0 \in \partial f(u_{k+1}) - A^T(\underbrace{\lambda_k + T_k(b - Au_{k+1} - Bv_k)}_{\hat{\lambda}_{k+1}}), \tag{5.141}$$

which is equivalent to $A^T\hat{\lambda}_{k+1} \in \partial f(u_{k+1})$. By exploiting properties of the Fenchel conjugate [Roc70], we get $u_{k+1} \in \partial f^*(A^T\hat{\lambda}_{k+1})$. A similar argument using the optimality condition for (3.2) leads to $v_{k+1} \in \partial g^*(B^T\lambda_{k+1})$. Recalling the definition of $\hat{f}, \hat{g}$ in (5.138), we arrive at

$$Au_{k+1} - b \in \partial \hat{f}(\hat{\lambda}_{k+1}) \quad \text{and} \quad Bv_{k+1} \in \partial \hat{g}(\lambda_{k+1}). \tag{5.142}$$

We can then use simple algebra to verify $\lambda_k, \hat{\lambda}_{k+1}$ in (3.3) and $\partial \hat{f}(\hat{\lambda}_{k+1}), \partial \hat{g}(\lambda_{k+1})$ in (5.142) satisfy the generalized DRS steps (5.139, 5.140). □

5.3.3 Generalized spectral stepsize rule

Proposition 4.2.1proved that the minimum residual of DRS can be obtained by setting the scalar penalty to $\tau_k = 1/\sqrt{\alpha\,\beta}$, where we assume the subgradients are locally linear as

$$\partial \hat{f}(\hat{\lambda}) = \alpha\,\hat{\lambda} + \Psi \qquad \text{and} \qquad \partial \hat{g}(\lambda) = \beta\,\lambda + \Phi, \tag{5.143}$$

$\alpha, \beta \in \mathbb{R}$ represent scalar curvatures, and $\Psi, \Phi \subset \mathbb{R}^p$.

We now present generalized spectral stepsize rules that can accomodate consensus problems.

Proposition 5.3.1 (Generalized spectral DRS)**.** *Suppose the generalized DRS steps (5.139, 5.140) are used, and assume the subgradients are locally linear,*

$$\partial \hat{f}(\hat{\lambda}) = M_\alpha\,\hat{\lambda} + \Psi \qquad \textit{and} \qquad \partial \hat{g}(\lambda) = M_\beta\,\lambda + \Phi. \tag{5.144}$$

for matrices $M_\alpha = diag(\alpha_1 I_d, \ldots, \alpha_N I_d)$ and $M_\beta = diag(\beta_1 I_d, \ldots, \beta_N I_d)$, and some $\Psi, \Phi \subset \mathbb{R}^p$. Then the minimal residual of $\hat{f}(\lambda_{k+1}) + \hat{g}(\lambda_{k+1})$ is obtained by setting $\tau_{i,k} = 1/\sqrt{\alpha_i\,\beta_i}$, $\forall i = 1, \ldots, N$.

Proof. Substituting subgradients $\partial \hat{f}(\hat{\lambda}), \partial \hat{g}(\lambda)$ into the generalized DRS steps

(5.139, 5.140), and using our linear assumption (5.144) yields

$$0 \in (T_k)^{-1}(\hat{\lambda}_{k+1} - \lambda_k) + (M_\alpha\, \hat{\lambda}_{k+1} + \Psi) + (M_\beta\, \lambda_k + \Phi)$$

$$0 \in (T_k)^{-1}(\lambda_{k+1} - \lambda_k) + (M_\alpha\, \hat{\lambda}_{k+1} + \Psi) + (M_\beta\, \lambda_{k+1} + \Phi).$$

Since T_k, M_α, M_β are diagonal matrices, we can split the equations into independent blocks, $\forall i = 1, \ldots, N$,

$$0 \in (\hat{\lambda}_{i,k+1} - \lambda_{i,k})/\tau_{i,k} + (\alpha_i\, \hat{\lambda}_{k+1} + \Psi_i) + (\beta_i\, \lambda_k + \Phi_i)$$

$$0 \in (\lambda_{i,k+1} - \lambda_{i,k})/\tau_{i,k} + (\alpha_i\, \hat{\lambda}_{k+1} + \Psi_i) + (\beta_i\, \lambda_{k+1} + \Phi_i).$$

Applying Proposition 5.3.1 in [Xu+17b] to each block, $\tau_{i,k} = 1/\sqrt{\alpha_i\, \beta_i}$ minimizes the block residual represented by $r^{DR}_{i,k} = \|(\alpha_i+\beta_i)\lambda_{k+1}+(a_i+b_i)\|$, where $a_i \in \Psi_i, b_i \in \Phi_i$. Hence the residual norm at step $k+1$, which is $\|(M_\alpha + M_\beta)\lambda_{k+1} + (a+b)\| = \sqrt{\sum_{i=1}^N (r^{DR}_{i,k+1})^2}$ is minimized by setting $\tau_{i,k} = 1/\sqrt{\alpha_i\, \beta_i}$, $\forall i = 1, \ldots, N$. □

5.3.4 Stepsize estimation for consensus problems

Thanks to the equivalence of ADMM and DRS, Proposition 5.3.1 can also be used to guide the selection of the "optimal" penalty parameter. We now show that the generalized spectral stepsizes can be estimated from the ADMM iterates for the primal consensus problem (5.132), without explicitly supplying the dual functions.

As in (5.142), the subgradients of dual functions $\partial \hat{f}$, $\partial \hat{g}$ can be computed from the ADMM iterates using the identities derived from (3.1, 3.2). For the consensus

problem we have $A = I_{dN}$, $B = -(I_d; \ldots; I_d)$, and $b = 0$, and so

$$(u_{1,k+1};\ \ldots;\ u_{N,k+1}) \in \partial \hat{f}(\hat{\lambda}_{k+1}) \tag{5.145}$$

$$-(\underbrace{v_{k+1};\ \ldots;\ v_{k+1}}_{N \text{ duplicates of } v_{k+1}}) \in \partial \hat{g}(\lambda_{k+1}). \tag{5.146}$$

If we approximate the behavior of these sub-gradients using the linear approximation (5.144), and break the sub-gradients into blocks (one for each worker node), we get (omitting iteration index k for clarity)

$$u_i = \alpha_i \hat{\lambda}_i + a_i \text{ and } -v = \beta_i \lambda_i + b_i,\ \forall i \tag{5.147}$$

where α_i and β_i represent the *curvature* of local functions $\hat{f}_i$ and $\hat{g}_i$ on the ith node.

We select stepsizes with a two step procedure, which follows the spectral stepsize literature. First, we estimate the local curvature parameters, α_i and β_i, by finding least-squares solutions to (5.147). Second, we plug these curvature estimates into the formula $\tau_{i,k} = 1/\sqrt{\alpha_i \beta_i}$. This formula produces the optimal stepsize when $\hat{f}$ and $\hat{g}$ are well approximated by a linear function, as shown in Proposition 5.3.1.

For notational convenience, we work with the quantities $\hat{\alpha}_{i,k} = 1/\alpha_i$, $\hat{\beta}_{i,k} = 1/\beta_i$, which are estimated on each node using the current iterates $u_{i,k}, v_k, \lambda_{i,k}, \hat{\lambda}_{i,k}$ and also an older iterate $u_{i,k_0}, v_{k_0}, \lambda_{i,k_0}, \hat{\lambda}_{i,k_0}, k_0 < k$. Defining $\Delta u_{i,k} = u_{i,k} - u_{i,k_0}$, $\Delta\hat{\lambda}_{i,k} = \hat{\lambda}_{i,k} - \hat{\lambda}_{i,k_0}$ and following the literature for Barzilai-Borwein/spectral stepsize estimation, there are two least squares estimators that can be obtained from (5.147):

$$\hat{\alpha}_{i,k}^{\mathrm{SD}} = \frac{\langle \Delta\hat{\lambda}_{i,k}, \Delta\hat{\lambda}_{i,k} \rangle}{\langle \Delta u_{i,k}, \Delta\hat{\lambda}_{i,k} \rangle} \text{ and } \hat{\alpha}_{i,k}^{\mathrm{SD}} = \frac{\langle \Delta u_{i,k}, \Delta\hat{\lambda}_{k} \rangle}{\langle \Delta u_{i,k}, \Delta u_{i,k} \rangle} \tag{5.148}$$

where SD stands for *steepest descent*, and MG stands for *minimum gradient*. [Zho+06] recommend using a hybrid of these two estimators, and choosing

$$\hat{\alpha}_{i,k} = \begin{cases} \hat{\alpha}_{i,k}^{\mathrm{MG}} & \text{if } 2\,\hat{\alpha}_{i,k}^{\mathrm{MG}} > \hat{\alpha}_{i,k}^{\mathrm{SD}} \\ \hat{\alpha}_{i,k}^{\mathrm{SD}} - \hat{\alpha}_{i,k}^{\mathrm{MG}}/2 & \text{otherwise.} \end{cases} \tag{5.149}$$

It was observed that this choice worked well for non-distributed ADMM in Chapter 4. We can similarly estimate $\hat{\beta}_{i,k}$ from $\Delta v_k = -v_k + v_{k_0}$ and $\Delta\lambda_{i,k} = \lambda_{i,k} - \lambda_{i,k_0}$.

ACADMM estimates the curvatures in the original d-dimensional feature space, and avoids estimating the curvature in the higher Nd-dimensional feature space (which grows with the number of nodes N in AADMM [Xu+17b]), which is especially useful for heterogeneous data with different distributions allocated to different nodes. The overhead of our adaptive scheme is only a few inner products, and the computation is naturally distributed on different workers.

5.3.5 Safeguarding and convergence

Spectral stepsizes for gradient descent methods are equipped with safeguarding strategies like backtracking line search to handle inaccurate curvature estimation and to guarantee convergence. To safeguard the proposed spectral penalty parameters, we check whether our linear subgradient assumption is reasonable before updating

Algorithm 4 Adaptive consensus ADMM (ACADMM)

Input: initialize v_0, $\lambda_{i,0}$, $\tau_{i,0}$, $k_0 = 0$,

while not converge by (5.137) **and** $k <$ maxiter **do**
  Locally update $u_{i,k}$ on each node by (5.133)
  Globally update v_k on central server by (5.134)
  Locally update dual variable $\lambda_{i,k}$ on each node by (5.135)
  if $\mathrm{mod}(k, T_f) = 1$ **then**
    Locally update $\hat{\lambda}_{i,k} = \lambda_{i,k-1} + \tau_{i,k}(v_{k-1} - u_{i,k})$
    Locally compute spectral stepsizes $\hat{\alpha}_{i,k}, \hat{\beta}_{i,k}$
    Locally estimate correlations $\alpha_{i,k}^{\mathrm{cor}}$, $\beta_{i,k}^{\mathrm{cor}}$
    Locally update $\tau_{i,k+1}$ using (5.151)
    $k_0 \leftarrow k$
  else
    $\tau_{i,k+1} \leftarrow \tau_{i,k}$
  end if
  $k \leftarrow k + 1$
end while

the stepsizes. We do this by testing that the correlations

$$\alpha_{i,k}^{\mathrm{cor}} = \frac{\langle \Delta u_{i,k}, \Delta\hat{\lambda}_{i,k} \rangle}{\|\Delta u_{i,k}\| \, \|\Delta\hat{\lambda}_{i,k}\|} \text{ and } \beta_{i,k}^{\mathrm{cor}} = \frac{\langle \Delta v_k, \Delta\lambda_{i,k} \rangle}{\|\Delta v_k\| \, \|\Delta\lambda_{i,k}\|}, \tag{5.150}$$

are bounded away from zero by a fixed threshold. We also bound changes in the penalty parameter by $(1+{}^{C^{\mathrm{cg}}}\!/_{k^2})$ according to Assumption 3.3.1, which was shown in Theorem 3.3.1 and Theorem 3.3.2 to guarantee convergence. The final safeguarded ACADMM rule is

$$\hat{\tau}_{i,k+1} = \begin{cases} \sqrt{\hat{\alpha}_{i,k}\hat{\beta}_{i,k}} & \text{if } \alpha_{i,k}^{\mathrm{cor}} > \epsilon^{\mathrm{cor}} \text{ and } \beta_{i,k}^{\mathrm{cor}} > \epsilon^{\mathrm{cor}} \\ \hat{\alpha}_{i,k} & \text{if } \alpha_{i,k}^{\mathrm{cor}} > \epsilon^{\mathrm{cor}} \text{ and } \beta_{i,k}^{\mathrm{cor}} \leq \epsilon^{\mathrm{cor}} \\ \hat{\beta}_{i,k} & \text{if } \alpha_{i,k}^{\mathrm{cor}} \leq \epsilon^{\mathrm{cor}} \text{ and } \beta_{i,k}^{\mathrm{cor}} > \epsilon^{\mathrm{cor}} \\ \tau_{i,k} & \text{otherwise,} \end{cases} \tag{5.151}$$

$$\tau_{i,k+1} = \max\{\min\{\hat{\tau}_{i,k+1}, \, (1 + \frac{C^{\mathrm{cg}}}{k^2})\tau_{i,k}\} \, , \, \frac{\tau_{i,k}}{1 + {}^{C^{\mathrm{cg}}}\!/_{k^2}}\}.$$

Application	Dataset	CADMM [Boy+11]	RB-ADMM [He+00]	AADMM [Xu+17b]	CRB-ADMM [Son+16]	**Proposed ACADMM**
Elastic net regression	Synthetic1	1000+(1.27e4)	94(1.22e3)	**43(563)**	106(1.36e3)	**48(623)**
	Synthetic2	1000+(1.27e4)	130(1.69e3)	341(4.38e3)	140(1.79e3)	**57(738)**
	MNIST	100+(1.49e4)	88(1.29e3)	40(5.99e3)	87(1.27e4)	**14(2.18e3)**
	CIFAR10 [2]	100+(1.04e3)	100+(1.06e3)	100+(1.05e3)	100+(1.05e3)	**35(376)**
	News20	100+(4.61e3)	100+(4.60e3)	100+(5.17e3)	100+(4.60e3)	**78(3.54e3)**
	RCV1	33(1.06e3)	31(1.00e3)	20(666)	31(1.00e3)	**8(284)**
	Realsim	32(5.91e3)	30(5.59e3)	14(2.70e3)	30(5.57e3)	**9(1.80e3)**
Sparse logistic regression	Synthetic1	138(137)	78(114)	80(101)	48(51.9)	**24(29.9)**
	Synthetic2	317(314)	247(356)	1000+(1.25e3)	1000+(1.00e3)	**114(119)**
	MNIST	325(444)	212(387)	325(516)	203(286)	**149(218)**
	CIFAR10	310(700)	152(402)	310(727)	149(368)	**44(118)**
	News20	316(4.96e3)	211(3.84e3)	316(6.36e3)	207(3.73e3)	**137(2.71e3)**
	RCV1	155(115)	155(116)	155(137)	155(115)	**150(114)**
	Realsim	184(77)	184(77)	184(85)	183(77)	**159(68)**
Support Vector Machine	Synthetic1	33(35.0)	33(49.8)	**19(27)**	26(28.4)	**21(25.3)**
	Synthetic2	283(276)	69(112)	1000+(1.59e3)	81(97.4)	**25(39.0)**
	MNIST	1000+(930)	172(287)	73(127)	285(340)	**41(88.0)**
	CIFAR10	1000+(774)	227(253)	231(249)	1000+(1.00e3)	**62(60.2)**
	News20	259(2.63e3)	262(2.74e3)	259(3.83e3)	267(2.78e3)	**217(2.37e3)**
	RCV1	47(21.7)	47(21.6)	47(31.1)	40(19.0)	**27(15.4)**
	Realsim	1000+(76.8)	1000+(77.6)	442(74.4)	1000+(79.3)	**347(41.6)**
SDP	Ham-9-5-6	100+(2.01e3)	100+(2.14e3)	35(860)	100+(2.14e3)	**30(703)**

Table 5.4: Iterations (and runtime in seconds);128 cores are used; absence of convergence after n iterations is indicated as n+.

5.3.6 Experiments & Applications

We now study the performance of ACADMM on benchmark problems, and compare to other methods.

Our experiments use the following test problems that are commonly solved using consensus methods.

Linear regression with elastic net regularizer. We consider consensus formulations of the elastic net [ZH05] with f_i and g defined as,

$$f_i(u_i) = \frac{1}{2}\|D_i u_i - c_i\|^2, \;\; g(v) = \rho_1|v| + \frac{\rho_2}{2}\|v\|^2, \tag{5.152}$$

where $D_i \in \mathbb{R}^{n_i \times m}$ is the data matrix on node i, and c_i is a vector of measurements.

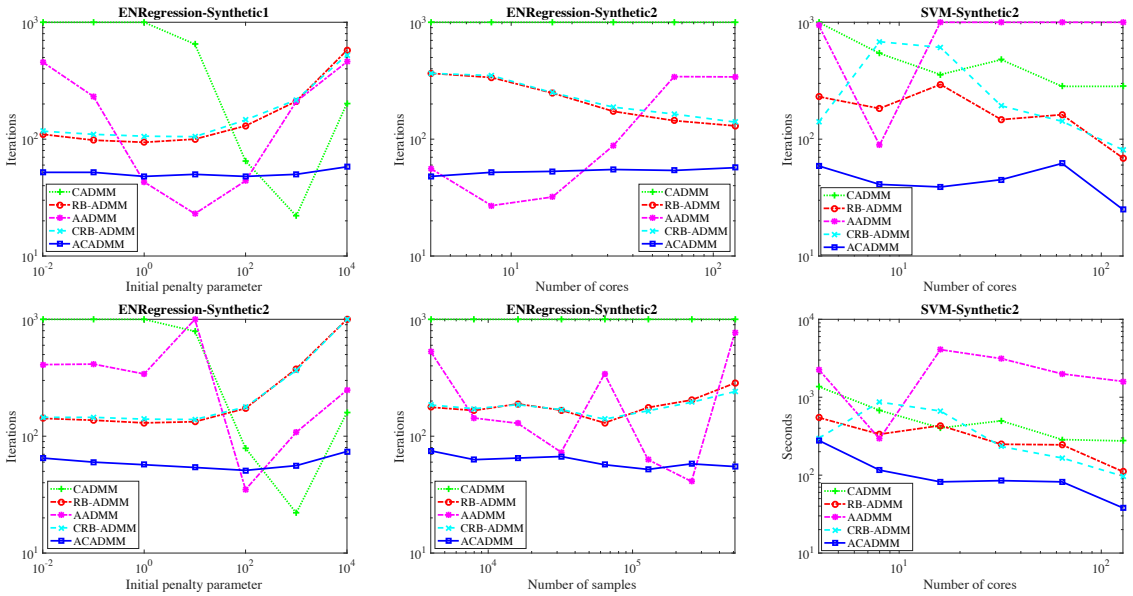
Sparse logistic regression with ℓ_1 regularizer can be written in the consensus form for distributed computing,

$$f_i(u_i) = \sum_{j=1}^{n_i} \log(1 + \exp(-c_{i,j} D_{i,j}^T u_i)), \;\; g(v) = \rho|v| \tag{5.153}$$

where $D_{i,j} \in \mathbb{R}^m$ is the jth sample, and $c_{i,j} \in \{-1, 1\}$ is the corresponding label. The minimization sub-step (5.133) in this case is solved by L-BFGS [LN89].

Support Vector Machines (SVMs) minimize the distributed objective function [Gol+16]

$$f_i(u_i) = C\sum_{j=1}^{n_i} \max\{1 - c_{i,j} D_{i,j}^T u_i, 0\}, \;\; g(v) = \frac{1}{2}\|v\|_2^2 \tag{5.154}$$



(a) Sensitivity of iteration count to initial penalty τ_0. Synthetic problems of EN regression are studied with 128 cores.

(b) Sensitivity of iteration count to number of cores (top) and number of samples (bottom).

(c) Sensitivity of iteration count (top) and wall time (bottom) to number of cores.

Figure 5.9: ACADMM is robust to the initial penalty τ, number of cores N, and number of training samples.

where $D_{i,j} \in \mathbb{R}^m$ is the jth sample on the ith node, and $c_{i,j} \in \{-1, 1\}$ is its label. The minimization (5.133) is solved by dual coordinate ascent [CL11].

Semidefinite programming (SDP) can be distributed as,

$$f_i(U_i) = \iota\{\mathcal{D}_i(U_i) = c_i\},\ g(v) = \langle F, V\rangle + \iota\{V \succeq 0\} \tag{5.155}$$

where $\iota\{S\}$ is a characteristic function that is 0 if condition S is satisfied and infinity otherwise. $V \succeq 0$ indicates that V is positive semidefinite. V, F, $D_{i,j} \in \mathbb{R}^{n\times n}$ are symmetric matrices, $\langle X, Y\rangle = \text{trace}(X^T Y)$ denotes the inner product of X and Y, and $\mathcal{D}_i(X) = (\langle D_{i,1}, X\rangle; \ldots; \langle D_{i,m_i}, X\rangle)$.

Experimental Setup We test these applications with synthetic and real datasets. *Synthetic1* contains samples from a normal distribution, and *Synthetic2* contains samples from a mixture of 10 random Gaussians. *Synthetic2* is heterogeneous because the data block on each individual node is sampled from only 1 of the 10 Gaussians. We also acquire large empirical datasets from the LIBSVM webpage [Liu+09], as well as MNIST digital images [LeC+98], and CIFAR10 object images [KH09]. For binary classification tasks (SVM and logreg), we equally split the 10 category labels of MNIST and CIFAR into "positive" and "negative" groups. We use a graph from the *Seventh DIMACS Implementation Challenge on Semidefinite and Related Optimization Problems* following [BM03] for Semidefinite Programming (SDP). The regularization parameter is fixed at $\rho = 10$ in all experiments.

Consensus ADMM (CADMM) [Boy+11], residual balancing (RB-ADMM) [He+00], adaptive ADMM (AADMM) [Xu+17b], and consensus residual balancing (CRB-ADMM) [Son+16] are implemented and reported for comparison. Hyper-parameters of these methods are set as suggested by their creators. The initial penalty is fixed at $\tau_0 = 1$ for all methods unless otherwise specified.

Convergence results Table 5.4 reports the convergence speed in iterations and wall-clock time (secs) for various test cases. These experiments are performed with 128 cores on a Cray XC-30 supercomputer. CADMM with default penalty $\tau = 1$ [Boy+11] is often slow to converge. ACADMM outperforms the other ADMM variants on all the real-world datasets, and is competitive with AADMM on two homogeneous synthetic datasets where the curvature may be globally estimated

with a scalar.

ACADMM is more reliable than AADMM since the curvature estimation becomes difficult for high dimensional variables. RB is relatively stable but sometimes has difficulty finding the exact optimal penalty, as the adaptation can stop because the difference of residuals are not significant enough to trigger changes. RB does not change the initial penalty in several experiments such as logistic regression on RCV1. CRB achieves comparable results with RB, which suggests that the relative sizes of local residuals may not always be very informative. ACADMM significantly boosts AADMM and the local curvature estimations are helpful in practice.

Robustness and sensitivity Fig. 5.9a shows that the practical convergence of ADMM is sensitive to the choice of penalty parameter. ACADMM is robust to the selection of the initial penalty parameter and achieves promising results for both homogeneous and heterogeneous data, comparable to ADMM with a fine-tuned penalty parameter.

We study scalability of the method by varying the number of workers and training samples (Fig. 5.9b). ACADMM is fairly robust to the scaling factor. AADMM occasionally performs well when small numbers of nodes are used, while ACADMM is much more stable. RB and CRB are more stable than AADMM, but cannot compete with ACADMM. Fig. 5.9c (bottom) presents the acceleration in (wall-clock secs) achieved by increasing the number of workers.

Finally, ACADMM is insensitive to the safeguarding hyper-parameters, correlation threshold ϵ^{cor} and convergence constant C^{cg}. Though tuning these parameters

may further improve the performance, the fixed default values generally perform well in our experiments and enable ACADMM to run without user oversight. In further experiments in the supplementary material, we also show that ACADMM is fairly insensitive to the regularization parameter ρ in our classification/regression models.

5.3.7 Summarization

We propose ACADMM, a fully automated algorithm for distributed optimization. Numerical experiments on various applications and real-world datasets demonstrate the efficiency and robustness of ACADMM. By automating the selection of algorithm parameters, adaptive methods make distributed systems more reliable, and more accessible to users that lack expertise in optimization.

5.4 Nonconvex Problems

5.4.1 Introduction

The alternating direction method of multipliers (ADMM) has been applied to solve a wide range of constrained convex and nonconvex optimization problems. ADMM decomposes complex optimization problems into sequences of simpler sub-problems that are often solvable in closed form. Furthermore, these sub-problems are often amenable to large-scale distributed computing environments [Gol+16; Tay+16]. ADMM solves the problem

$$\min_{u\in\mathbb{R}^n, v\in\mathbb{R}^m} H(u) + G(v), \quad \text{subject to} \quad Au + Bv = b, \tag{5.156}$$

where $H : \mathbb{R}^n \to \bar{\mathbb{R}}$, $G : \mathbb{R}^m \to \bar{\mathbb{R}}$, $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$, and $b \in \mathbb{R}^p$, by the following steps,

$$u_{k+1} = \arg\min_u H(u) + \langle \lambda_k, -Au \rangle + \frac{\tau_k}{2} \|b - Au - Bv_k\|_2^2 \tag{5.157}$$

$$v_{k+1} = \arg\min_v G(v) + \langle \lambda_k, -Bv \rangle + \frac{\tau_k}{2} \|b - Au_{k+1} - Bv\|_2^2 \tag{5.158}$$

$$\lambda_{k+1} = \lambda_k + \tau_k(b - Au_{k+1} - Bv_{k+1}), \tag{5.159}$$

where $\lambda \in \mathbb{R}^p$ is a vector of dual variables (Lagrange multipliers), and τ_k is a scalar penalty parameter.

The convergence of the algorithm can be monitored using primal and dual "residuals," both of which approach zero as the iterates become more accurate, and which are defined as

$$r_k = b - Au_k - Bv_k, \qquad \text{and} \qquad d_k = \tau_k A^T B(v_k - v^{k-1}), \tag{5.160}$$

respectively [Boy+11]. The iteration is generally stopped when

$$\|r_k\|_2 \leq \epsilon^{tol} \max\{\|Au_k\|_2, \|Bv_k\|_2, \|b\|_2\} \quad \text{and} \quad \|d_k\|_2 \leq \epsilon^{tol} \|A^T \lambda_k\|_2, \tag{5.161}$$

where $\epsilon^{tol} > 0$ is the stopping tolerance.

ADMM was introduced by Glowinski and Marroco [GM75] and Gabay and Mercier [GM76], and convergence has been proved under mild conditions for convex problems [Gab83; EB92; HY15]. The practical performance of ADMM on convex

problems has been extensively studied, see [Boy+11; Gol+14b; Xu+17b] and references therein. For nonconvex problems, the convergence of ADMM under certain assumptions are studied in [Wan+14; LP15; Hon+16; Wan+15]. The current weakest assumptions are given in [Wan+15], which requires a number of strict conditions on the objective, including a Lipschitz differentiable objective term. In practice, ADMM has been applied on various nonconvex problems, including nonnegative matrix factorization [Xu+12], ℓ_p-norm regularization $(0 < p < 1)$[Bou+13; CW13], tensor factorization [LS15; Xu+16b], phase retrieval [Wen+12], manifold optimization [LO14; Kov+15], random fields [Mik+14], and deep neural networks [Tay+16].

The penalty parameter τ_k is the only free choice in ADMM, and plays an important role in the practical performance of the method. Adaptive methods have been proposed to automatically tune this parameter as the algorithm runs. The residual balancing method [He+00] automatically increase or decrease the penalty so that the primal and dual residuals have approximately similar magnitudes. The more recent AADMM method [Xu+17b] uses a spectral (Barzilai-Borwein) rule for tuning the penalty parameter. These methods achieve impressive practical performance for convex problems and are guaranteed to converge under moderate conditions (such as when adaptivity is stopped after a finite number of iterations).

In this manuscript, we study the practical performance of ADMM on several nonconvex applications, including ℓ_0 regularized linear regression, ℓ_0 regularized image denoising, phase retrieval, and eigenvector computation. While the convergence of these applications may (not) be guaranteed by the current theory, ADMM is one of the (popular) choices to solve these nonconvex problems. The following questions

are addressed using these model problems: (i) does ADMM converge in practice, (ii) does the update order of $H(u)$ and $G(v)$ matter, (iii) is the local optimal solution good, (iv) does the penalty parameter τ_k matter, and (v) is an adaptive penalty choice effective?

5.4.2 Nonconvex applications

ℓ_0 regularized linear regression. Sparse linear regression can be achieved using the non-convex, ℓ_0 regularized problem

$$\min_x \frac{1}{2}\|Dx - c\|_2^2 + \rho\|x\|_0, \tag{5.162}$$

where $D \in \mathbb{R}^{n\times m}$ is the data matrix, c is a measurement vector, and x is the regression coefficients. ADMM is applied to solve problem (5.162) using the equivalent formulation

$$\min_{u,v} \frac{1}{2}\|Du - c\|_2^2 + \rho\|v\|_0 \quad \text{subject to} \quad u - v = 0. \tag{5.163}$$

ℓ_0 regularized image denoising. The ℓ_0 regularizer [DZ13] can be substituted for the ℓ_1 regularizer when computing total variation for image denoising. This results in the formulation [Cha07]

$$\min_x \frac{1}{2}\|x - c\|_2^2 + \rho\|\nabla x\|_0 \tag{5.164}$$

where c represents a given noisy image, ∇ is the linear discrete gradient operator, and $\|\cdot\|_2/\|\cdot\|_0$ is the ℓ_2/ℓ_0 norm. We solve the equivalent problem

$$\min_{u,v} \frac{1}{2}\|u-c\|_2^2 + \rho\|v\|_0 \quad \text{subject to} \quad \nabla u - v = 0. \tag{5.165}$$

The resulting ADMM sub-problems can be solved in closed form using fast Fourier transforms [GO09].

Phase retrieval. Ptychographic phase retrieval [Yan+11; Wen+12] solves the problem

$$\min_{x} \frac{1}{2}||\text{abs}(Dx) - c||_2^2, \tag{5.166}$$

where $x \in \mathbb{C}^n$, $D \in \mathbb{C}^{m\times n}$, and $\text{abs}(\cdot)$ denotes the elementwise magnitude of a complex vector. ADMM is applied to the equivalent problem

$$\min_{u,v} \frac{1}{2}||\text{abs}(u) - c||_2^2 \;\text{ subject to }\; u - Dv = 0. \tag{5.167}$$

Eigenvector problem. The eigenvector problem is a fundamental problem in numerical linear algebra. The leading eigenvalue of a matrix D is found by computing

$$\max \|Dx\|_2^2 \quad \text{subject to} \quad \|x\|_2 = 1. \tag{5.168}$$

ADMM is applied to the equivalent problem

$$\min -\|Du\|_2^2 + \iota_{\{z:\,\|z\|_2=1\}}(v) \quad \text{subject to} \quad u - v = 0, \tag{5.169}$$

where ι_S is the characteristic function defined by $\iota_S(v) = 0$, if $v \in S$, and $\iota_S(v) = \infty$, otherwise.

5.4.3 Experiments & Observations

Experimental setting. We implemented "vanilla ADMM" (ADMM with constant penalty), and fast ADMM with Nesterov acceleration and restart [Gol+14b]. We also implemented two methods for automatically selecting penalty parameters: residual balancing [He+00], and the spectral adaptive method [Xu+17b]. For ℓ_0 regularized linear regression, the synthetic problem in [ZH05; Gol+14b; Xu+17b] and realistic problems in [Efr+04; ZH05; Xu+17b] are investigated with $\rho = 1$. For ℓ_0 regularized image denoising, a one-dimensional synthetic problem was created by the process described in [ZH05], and is shown in Fig. 5.12. For the total-variation experiments, the "Barbara" , "Cameraman", and "Lena" images are investigated, where Gaussian noise with zero mean and standard deviation 20 was added to each image (Fig. 5.13). $\rho = 1$ and $\rho = 500$ are used for the synthetic problem and image problems, respectively. For phase retrieval, a synthetic problem is constructed with a random matrix $D \in \mathbb{C}^{15000\times 500}$, $x \in \mathbb{C}^{500}$, $e \in \mathbb{C}^{15000}$ and $c = \text{abs}(Dx + e)$. Three images in Fig. 5.13 are used. Each image is measured with 21 octanary pattern filters as described in [Can+15]. For the eigenvector problem,

a random matrix $D \in \mathbb{R}^{20\times 20}$ is used.

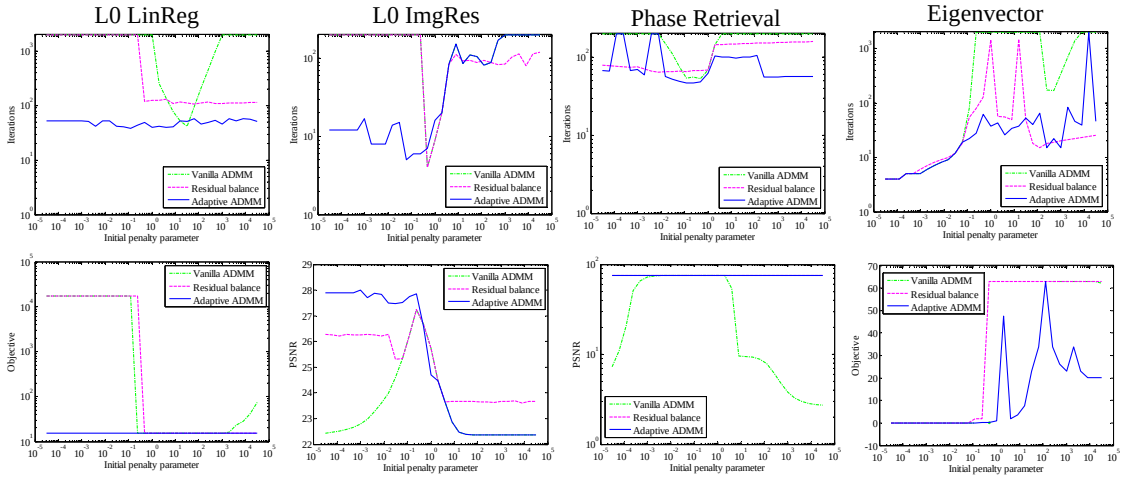


Figure 5.10: Sensitivity to the (initial) penalty parameter τ_0 for the ℓ_0 regularized linear regression, eigenvector computation, "cameraman" denoising, and phase retrieval. (top) Number of iterations needed as a function of initial penalty parameter. (bottom) The objective/PSNR of the minima found for each non-convex problem.

Does ADMM converge in practice? The convergence of vanilla ADMM is quite sensitive to the choice of penalty parameter. For vanilla ADMM, the iterates may oscillate, and if convergence occurs it may be very slow when the penalty parameter is not properly tuned. The residual balancing method converges more often than vanilla ADMM, and the spectral adaptive ADMM converges the most often. However, none of these methods uniformly beats all others, and it appears that vanilla ADMM with a highly tuned stepsize can sometimes outperform adaptive variants.

Does the update order of $H(u)$ and $G(v)$ matter? In Fig. 5.10, ADMM is performed by first minimizing with respect to the smooth objective term, and then the nonsmooth term. We repeat the experiments with the update order swapped, and report the results in Fig. 5.11 of the appendix. When updating the non-smooth

term first, the convergence of ADMM for the phase retrieval problem becomes less reliable. However, for some problems (like image denoising), convergence happened a bit faster than with the original update order. Although the behavior of ADMM changes, there is no predictable difference between the two update orderings.

Is the local optimal solution good? The bottom row of Fig. 5.10 presents the objective/PSNR achieved by the ADMM variants when varying the (initial) penalty parameter. In general, the quality of the solution depends strongly on the penalty parameter chosen. There does not appear to be a predictable relationship between the best penalty for convergence speed and the best penalty for solution quality.

Does the adaptive penalty work? In Table 5.5, we see that adaptivity not only speeds up convergence, but for most problem instances it also results in better minimizers. This behavior is not uniform across all experiments though, and for some problems a slightly lower objective value can be achieved using a finely tuned constant stepsize.

5.4.4 Appendix: implementation details

5.4.4.1 ℓ_0 regularized linear regression

ℓ_0 regularized linear regression is a nonconvex problem

$$\min_x \frac{1}{2}\|Dx - c\|_2^2 + \rho\|x\|_0 \tag{5.170}$$

Application	Dataset	#samples × #features[1]	Vanilla ADMM	Residual balance [He+00]	**Adaptive ADMM [Xu+17b]**
ℓ_0 regularized linear regression	Synthetic	50 × 40	2000+(.621) 1.71e4	2000+(.604) 1.71e4	**39(.018)** **15.2**
	Boston	506 × 13	2000+(.598) 1.50e5	2000+(.570) 1.50e5	**1039(.342)** **1.34e5**
	Diabetes	768 × 8	2000+(.751) 384	2000+(.708) 648	**28(.014)** **285**
	Leukemia	38 × 7129	2000+(15.3) 19.0	78(.578) 19.0	**63(.477)** **19.0**
	Prostate	97 × 8	2000+(.413) 1.14e3	2000+(.466) 380	**29(.013)** **324**
	Servo	130 × 4	2000+(.426) 267	2000+(.471) 267	**45(.014)** **198**
ℓ_0 regularized image restoration	Synthetic1D	100 × 1	2000+(.701) 40.6	1171(.409) 45.4	**866(.319)** **45.4**
	Barbara	512 × 512	200+(35.5) 24.7	200+(35.1) 24.7	**18(3.33)** **24.7**
	Cameraman	256 × 256	200+(5.75) 25.9	200+(5.60) 25.9	**6(.190)** **27.8**
	Lena	512 × 512	200+(35.5) 25.9	200+(35.8) 25.9	**11(1.98)** **27.9**
phase retrieval	Synthetic	15000 × 500	200+(19.4)	94(9.01)	**46(4.45)**
	Barbara	512 × 512 × 21	59(91.1) 81.5	59(89.6) 81.5	**50(88.1)** **81.5**
	Cameraman	256 × 256 × 21	59(29.6) 75.7	55(19.4) 75.7	**48(20.8)** **75.7**
	Lena	512 × 512 × 21	59(90.1)) 81.4	57(87.4) 81.5	**52(92.0)** **81.5**

[1] width × height for image restoration; width × height × filters for phase retrieval

Table 5.5: Iterations (with runtime in seconds) and objective (or PSNR) for the various algorithms and applications described in the text. Absence of convergence after n iterations is indicated as $n+$.

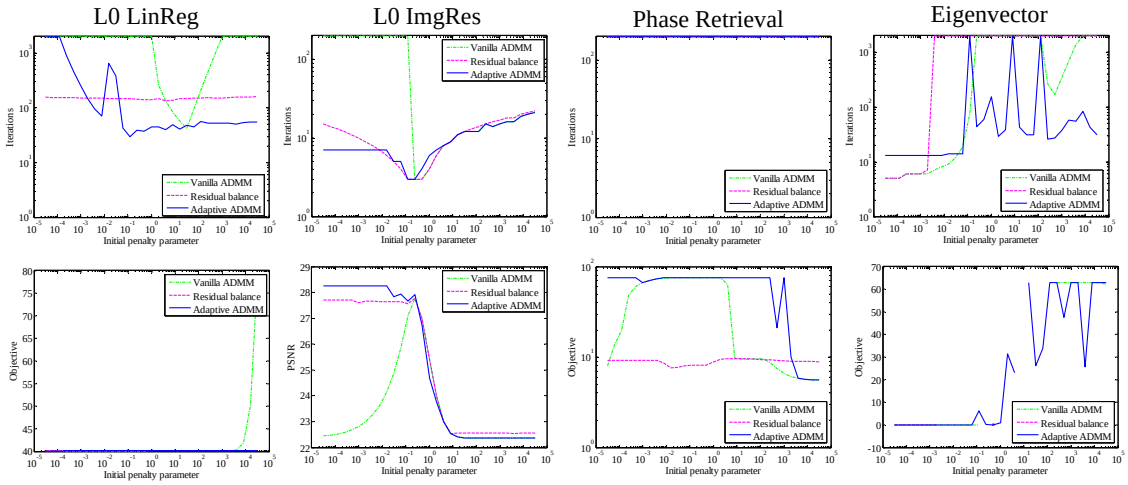


Figure 5.11: Convergence results when the non-smooth objective term is updated first, and the smooth term is updated second. Sensitivity to the (initial) penalty parameter τ_0 is shown for the synthetic problem of ℓ_0 regularized linear regression, eigenvector computation, the "cameraman" denoising problem, and phase retrieval. The top row shows the convergence speed in iterations. The bottom row shows the objective/PSNR achieved by the final iterates.

where $D \in \mathbb{R}^{n\times m}$ is the data matrix, c is the measurement vector, and x is the regression coefficients. ADMM is applied to solve problem (5.170) by solving the equivalent problem

$$\min_{u,v} \frac{1}{2}\|Du - c\|_2^2 + \rho\|v\|_0 \quad \text{subject to} \quad u - v = 0. \tag{5.171}$$

The proximal operator of the ℓ_0 norm is the hard-thresholding,

$$\text{hard}(z,t) = \arg\min_x \|x\|_0 + \frac{1}{2t}\|x - z\|_2^2 = z \odot \mathcal{I}_{\{z:|z|>\sqrt{2t}\}}(z), \tag{5.172}$$

where $\odot$ represents element-wise multiplication, and $\mathcal{I}_S$ is the indicator function of the set S: $\mathcal{I}_S(v) = 1$, if $v \in S$, and $\mathcal{I}_S(v) = 0$, otherwise. Then the steps of ADMM

can be written

$$u_{k+1} = \arg\min_u \|Du - c\|_2^2 + \frac{\tau}{2}\|0 - u + v_k + \lambda_k/\tau\|_2^2 \tag{5.173}$$

$$= \begin{cases} (D^T D + \tau I_n)^{-1}(\tau v_k + \lambda_k + D^T c) & \text{if } n \geq m \\ (I_n - D^T(\tau I_m + DD^T)^{-1}D)(v_k + \lambda_k/\tau + D^T c/\tau) & \text{if } n < m \end{cases} \tag{5.174}$$

$$v_{k+1} = \arg\min_v \rho\|v\|_0 + \frac{\tau}{2}\|0 - u_{k+1} + v + \lambda_k/\tau\|_2^2 \tag{5.175}$$

$$= \text{hard}(u_{k+1} - \lambda_k/\tau, \rho/\tau) \tag{5.176}$$

$$\lambda_{k+1} = \lambda_k + \tau(0 - u_{k+1} + v_{k+1}). \tag{5.177}$$

5.4.4.2 ℓ_0 regularized image denoising

The ℓ_0 regularizer [DZ13] is an alternative to the ℓ_1 regularizer when computing total variation [GO09; Gol+14b]. ℓ_0 regularized image denoising solves the nonconvex problem

$$\min_x \frac{1}{2}\|x - c\|_2^2 + \rho\|\nabla x\|_0 \tag{5.178}$$

where c represents a given noisy image, ∇ is the linear gradient operator, and $\|\cdot\|_2/\|\cdot\|_0$ denotes the ℓ_2/ℓ_0 norm of vectors. The steps of ADMM for this problem

are

$$u_{k+1} = \arg\min_u \frac{1}{2}\|u - c\|_2^2 + \frac{\tau}{2}\|v_k + \lambda_k/\tau - \nabla u\|_2^2 \tag{5.179}$$

$$= (I + \tau\nabla^T\nabla)^{-1}(c + \tau\nabla^T(v_k + \lambda_k/\tau)) \tag{5.180}$$

$$v_{k+1} = \arg\min_v \rho\|v\|_0 + \frac{\tau}{2}\|0 - \nabla u_{k+1} + v + \lambda_k/\tau\|^2 \tag{5.181}$$

$$= \text{hard}(\nabla u_{k+1} - \lambda_k/\tau, \rho/\tau) \tag{5.182}$$

$$\lambda_{k+1} = \lambda_k + \tau(0 - \nabla u_{k+1} + v_{k+1}) \tag{5.183}$$

where the linear systems can be solved using fast Fourier transforms.

5.4.4.3 Phase retrieval

Ptychographic phase retrieval [Yan+11; Wen+12] solves problem

$$\min_x \frac{1}{2}||\text{abs}(Dx) - c||_2^2, \tag{5.184}$$

where $x \in \mathbb{C}^n$, $D \in \mathbb{C}^{m\times n}$, and $\text{abs}(\cdot)$ denotes the elementwise magnitude of a complex-valued vector. ADMM is applied to the equivalent problem

$$\min_{u,v} \frac{1}{2}||\text{abs}(u) - c||_2^2 \;\text{ subject to }\; u - Dv = 0. \tag{5.185}$$

Define the projection operator of a complex valued vector as

$$\begin{aligned} \text{absProj}(z, c, t) &= \min_x \tfrac{1}{2}\|\text{abs}(x) - c\|_2^2 + \tfrac{t}{2}\|x - z\|_2^2 && (5.186) \\ &= \left(\tfrac{t}{1+t}\,\text{abs}(z) + \tfrac{1}{1+t}\,c\right) \odot \text{sign}(z), && (5.187) \end{aligned}$$

where $\text{sign}(\cdot)$ denotes the elementwise phase of a complex-valued vector. In the following ADMM steps, notice that the dual variable $\lambda \in \mathbb{C}^m$ is complex, and the penalty parameter $\tau \in \mathbb{R}$ is a real non-negative scalar,

$$\begin{aligned} u_{k+1} &= \arg\min_u \frac{1}{2}\|\text{abs}(u) - c\|_2^2 + \frac{\tau}{2}\|Dv_k + \lambda_k/\tau - u\|_2^2 && (5.188) \\ &= \text{absProj}(Dv_k + \lambda_k/\tau, c, \tau) && (5.189) \\ v_{k+1} &= \arg\min_v\ 0 + \frac{\tau}{2}\|0 - u_{k+1} + Dv + \lambda_k/\tau\|_2^2 = D^{-1}(u_{k+1} - \lambda_k/\tau) && (5.190) \\ \lambda_{k+1} &= \lambda_k + \tau(0 - u_{k+1} + Dv_{k+1}). && (5.191) \end{aligned}$$

5.4.4.4 Eigenvector problem

The eigenvector problem is a fundamental problem in numerical linear algebra. The leading eigenvector of a matrix can be recovered by solving the Rayleigh quotient maximization problem

$$\max \|Dx\|_2^2 \quad \text{subject to} \quad \|x\|_2 = 1. \tag{5.192}$$

ADMM is applied to the equivalent problem

$$\min -\|Du\|_2^2 + \iota_{\{z:\,\|z\|_2=1\}}(v) \quad \text{subject to} \quad u - v = 0, \tag{5.193}$$

where ι_S is the characteristic function of the set S: $\iota_S(v) = 0$, if $v \in S$, and $\iota_S(v) = \infty$, otherwise. The ADMM steps are

$$u_{k+1} = \arg\min_u -\|Du\|_2^2 + \frac{\tau}{2}\|0 - u + v_k + \lambda_k/\tau\|_2^2 = (\tau I - 2D^T D)^{-1}(\tau v_k + \lambda_k) \tag{5.194}$$

$$v_{k+1} = \arg\min_v \iota_{\{z:\,\|z\|_2=1\}}(v) + \frac{\tau}{2}\|0 - u_{k+1} + v + \lambda_k/\tau\|^2 = \frac{u_{k+1} - \lambda_k/\tau}{\|u_{k+1} - \lambda_k/\tau\|_2} \tag{5.195}$$

$$\lambda_{k+1} = \lambda_k + \tau(0 - u_{k+1} + v_{k+1}). \tag{5.196}$$

5.4.5 Appendix: synthetic and realistic datasets

We provide the detailed construction of the synthetic dataset for our linear regression experiments. The same synthetic dataset has been used in [ZH05; Gol+14b; Xu+17b]. Based on three random normal vectors $\nu_a, \nu_b, \nu_c \in \mathbb{R}^{50}$, the data matrix

$D = [d_1 \dots d_{40}] \in \mathbb{R}^{50\times 40}$ is defined as

$$d_i = \begin{cases} \nu_a + e_i, & i = 1, \dots, 5, \\ \nu_b + e_i, & i = 6, \dots, 10, \\ \nu_c + e_i, & i = 11, \dots, 15, \\ \nu_i \in N(0,1), & i = 16, \dots, 40, \end{cases} \tag{5.197}$$

where e_i are random normal vectors from $N(0,1)$. The problem is to recover the vector

$$x^* = \begin{cases} 3, & i = 1, \dots, 15, \\ 0, & \text{otherwise} \end{cases} \tag{5.198}$$

from noisy measurements of the form $c = Dx^* + \hat{e}$, with $\hat{e} \in N(0, 0.1)$

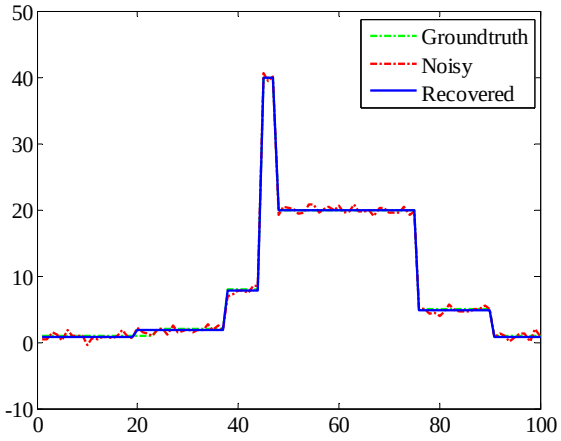


Figure 5.12: The synthetic one-dimensional signal for ℓ_0 regularized image denoising. The groundtruth signal, noisy signal (PSNR = 37.8) and recovered signal by AADMM (PSNR = 45.4) are shown.

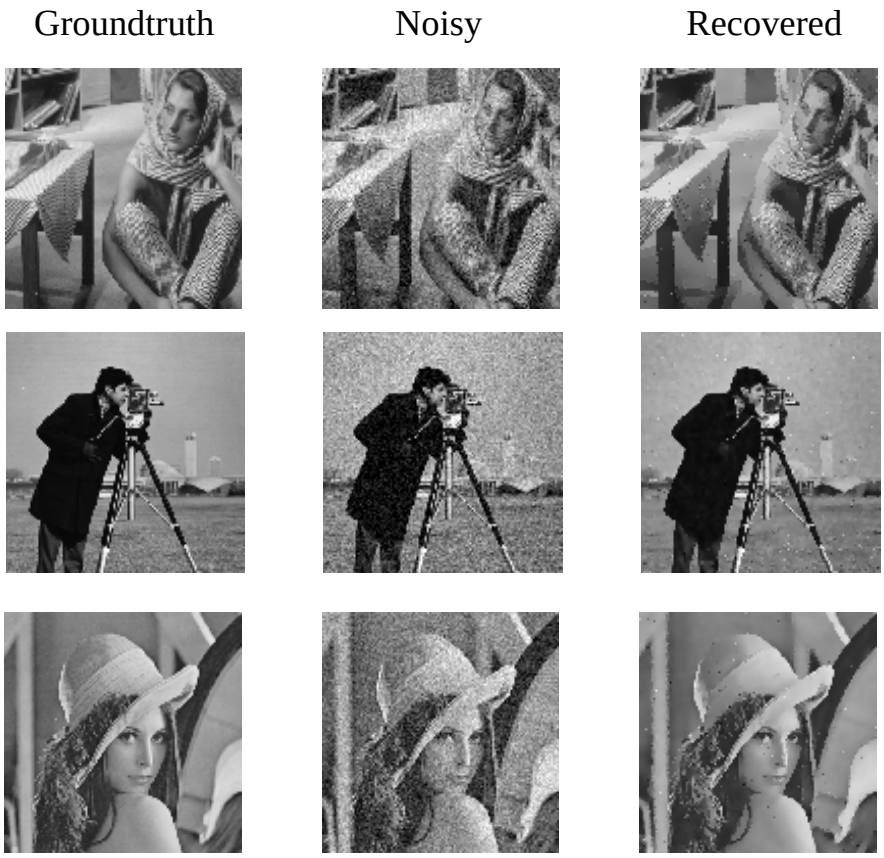


Figure 5.13: The groundtruth image (left), noisy image (middle), and recovered image by AADMM (right) for ℓ_0 regularized image denoising. The PSNR of the noisy/recovered images are 21.9/24.7 for "Barbara", 22.4/27.8 for "Cameraman", 21.9/27.9 for "Lena".

5.4.6 Summarization

We provide a detailed discussion of the performance of ADMM on several non-convex applications, including ℓ_0 regularized linear regression, ℓ_0 regularized image denoising, phase retrieval, and eigenvector computation. In practice, ADMM usually converges for those applications, and the penalty parameter choice has a significant effect on both convergence speed and solution quality. Adaptive penalty methods such as AADMM [Xu+17b] automatically select the penalty parameter, and perform optimization with little user oversight. For most problems, adaptive stepsize methods result in faster convergence or better minimizers than vanilla ADMM with a constant non-tuned penalty parameter. However, for some difficult non-convex problems, the best results can still be obtained by fine-tuning the penalty parameter.

Part II

GAN, Network Acceleration and Image Processing

Chapter 6: Stochastic Alternating Methods

We have shown how to solve min-max saddle point problem derived from constrained problem. We also show how to distribute large scale problem on a large number of computing nodes. Now let us consider the more general min-max problem and how we can efficiently optimize it by stochastic method without distributed computing. In this chapter, we review our theoretical insights on stochastic alternating method and prediction method, and refer readers to [Yad+18] for empirical performance of applying prediction step to stabalize generative adversarial network (GAN) training. In the following chapters, we will apply GAN for image processing in Chapter 7, and GAN for network acceleration in Chapter 8.

6.1 Stochastic Alternating Methods with Prediction Step

Let us consider optimizing the minimax problem that is more general than the Lagrangian saddle point problem (2.2),

$$\min_u \max_v \mathcal{L}(u, v). \tag{6.1}$$

An attractive application of the minimax problem is the training of generative adversarial networks (GANs).

Deep neural networks achieve state-of-the-art performance in many machine learning applications. Despite the complicated network architectures and the non-convexity in the training objective, the relatively simple solver, stochastic gradient descent (SGD), often performs well in practice. Though it is not fully understood why SGD works so well in practice, recent theoretical study [Cho+15; Kaw16; Din+17; Kaw+17; San+19] and empirical study [Goo+14b; Kes+16; Cha+16b; Li+18b] provided a lot of insightful explanation. We consider the stochastic version of the alternating optimization methods for general minimax problems (6.1). One of the appealing applications of this method is training generative neural networks (GANs). GANs have been extensively studied over recent years, and we recently applied GANs for accelerating neural networks [Xu+18b], transferring image styles [Xu+19a], and dehazing images [Yan+18], andin computer vision. The stochastic alternating gradients method for training GANs can be written as

$$u_{k+1} = u^k - \tau_k \mathcal{L}'_u(u^k, v^k) \quad | \quad \text{gradient descent in } u \tag{6.2}$$

$$v_{k+1} = v^k + \sigma_k \mathcal{L}'_v(\bar{u}_{k+1}, v^k) \quad | \quad \text{gradient ascent in } v, \tag{6.3}$$

where $\{\tau_k, \sigma_k\}$ are the stepsizes.

The training of GANs is not well understood and is also known to be unstable. We showed that the stochastic alternating methods for training GANs can be stabilized by introducing the prediction step inspired by the primal-dual gradient

methods [CP11; Gol+15], which enables users to choose bigger stepsizes and achieve better models [Yad+18],

$$u_{k+1} = u^k - \tau_k \mathcal{L}'_u(u^k, v^k) \quad | \quad \text{gradient descent in } u \tag{6.4}$$

$$\bar{u}_{k+1} = u_{k+1} + (u_{k+1} - u^k) \quad | \quad \textit{predict} \text{ future value of } u \tag{6.5}$$

$$v_{k+1} = v^k + \sigma_k \mathcal{L}'_v(\bar{u}_{k+1}, v^k) \quad | \quad \text{gradient ascent in } v. \tag{6.6}$$

6.2 Background and Advantage of Prediction Step

In the convex optimization literature, saddle point problems are well studied. One popular solver is the primal-dual hybrid gradient (PDHG) method [ZC08; Ess+09], which has been popularized by Chambolle and Pock [CP11], and has been successfully applied to a range of machine learning and statistical estimation problems [Gol+15]. PDHG relates closely to the method proposed here - it achieves stability using the same prediction step, although it uses a different type of gradient update and is only applicable to bi-linear problems.

Stochastic methods for convex saddle-point problems can be roughly divided into two categories: stochastic coordinate descent [DL14; LZ15; ZL15; ZS15; ZS16; WX17; ST17] and stochastic gradient descent [Che+14; Qia+16]. Similar optimization algorithms have been studied for reinforcement learning [WC16; Du+17]. Recently, a "doubly" stochastic method that randomizes both primal and dual updates was proposed for strongly convex bilinear saddle point problems [Yu+15]. For general saddle point problems, "doubly" stochastic gradient descent methods

are discussed in Nemirovski et al. [Nem+09],Palaniappan and Bach [PB16], in which primal and dual variables are updated simultaneously based on the previous iterates and the current gradients.

However, stochastic alternating method can be unstable and inaccurate. Let us look at the behavior of (6.2)-(6.3) on a simple bi-linear saddle of the form

$$\mathcal{L}(u, v) = v^T K u \tag{6.7}$$

where K is a matrix. When exact (non-stochastic) gradient updates are used, the iterates follow the path of a simple dynamical system with closed-form solutions. We give here a sketch of this argument, and show that, as the learning rate gets small, the iterates of the non-prediction method rotate in orbits around the saddle without converging, while the iterates of the prediction method converge.

When the (non-predictive) gradient method is applied to the linear problem (6.7), the resulting iterations can be written

$$\frac{u_{k+1} - u^k}{\alpha} = -K^T v^k, \qquad \frac{v_{k+1} - v^k}{\alpha} = (\beta/\alpha) K u_{k+1}.$$

When the stepsize α gets small, this behaves like a discretization of the system of differential equations

$$\dot{u} = -K^T v, \qquad \dot{v} = \beta/\alpha K u$$

where $\dot{u}$ and $\dot{v}$ denote the derivatives of u and v with respect to time. These

equations describe a simple harmonic oscillator, and the closed form solution for u is

$$u(t) = C\cos(\Sigma^{1/2}t + \phi)$$

where Σ is a diagonal matrix, and the matrix C and vector ϕ depend on the initialization. We can see that, for small values of α and β, the non-predictive algorithm approximates an undamped harmonic motion, and the solutions orbit around the saddle without converging.

The prediction step (6.5) improves convergence because it produces *damped* harmonic motion that sinks into the saddle point. When applied to the linearized problem (6.7), we get the dynamical system

$$\dot{u} = -K^T v, \qquad \dot{v} = \beta/\alpha K(u + \alpha\dot{u}) \tag{6.8}$$

which has solution

$$u(t) = UA\exp(-\frac{t\alpha}{2}\sqrt{\Sigma})\sin(t\sqrt{(1-\alpha^2/4)\Sigma} + \phi).$$

From this analysis, we see that the damping caused by the prediction step causes the orbits to converge into the saddle point, and the error decays exponentially fast.

6.3 Convergence for Convex-concave Problem

In this section, we prove that for convex-concave saddle point problems, the above stochastic steps have worst-case $O(1/\sqrt{k})$ convergence rate, and the prediction

step stabilizes the optimization in theory.

We assume that the function $\mathcal{L}(u, v)$ is convex in u and concave in v. We can then measure convergence using the "primal-dual" gap, $P(u, v) = \mathcal{L}(u, v^\star) - \mathcal{L}(u^\star, v)$ where $(u^\star, v^\star)$ is a saddle. Note that $P(u, v) > 0$ for non-optimal (u, v), and $P(u, v) = 0$ if (u, v) is a saddle. Using these definitions, we formulate the following convergence result. The proof is in the supplementary material.

Theorem 6.3.1. *Suppose the function $\mathcal{L}(u, v)$ is convex in u, concave in v, and that the partial gradient $\mathcal{L}'_v$ is uniformly Lipschitz smooth in u ($\|\mathcal{L}'_v(u_1, v) - \mathcal{L}'_v(u_2, v)\| \leq L_v\|u_1 - u_2\|$). Suppose further that the stochastic gradient approximations satisfy $\mathbb{E}\|\mathcal{L}'_u(u, v)\|^2 \leq G_u^2$, $\mathbb{E}\|\mathcal{L}'_v(u, v)\|^2 \leq G_v^2$ for scalars G_u and G_v, and that $\mathbb{E}\|u^k - u^\star\|^2 \leq D_u^2$, and $\mathbb{E}\|v^k - v^\star\|^2 \leq D_v^2$ for scalars D_u and D_v.*

If we choose decreasing learning rate parameters of the form $\alpha_k = \frac{C_\alpha}{\sqrt{k}}$ and $\beta_k = \frac{C_\beta}{\sqrt{k}}$, then the SGD method with prediction converges in expectation, and we have the error bound

$$\mathbb{E}[P(\hat{u}^l, \hat{v}^l)] \leq \frac{1}{2\sqrt{l}}\left(\frac{D_u^2}{C_\alpha} + \frac{D_v^2}{C_\beta}\right) + \frac{\sqrt{l+1}}{l}\left(\frac{C_\alpha G_u^2}{2} + C_\alpha L_v G_u^2 + C_\alpha L_v D_v^2 + \frac{C_\beta G_v^2}{2}\right)$$

where $\hat{u}^l = \frac{1}{l}\sum_{k=1}^{l} u^k$, $\hat{v}^l = \frac{1}{l}\sum_{k=1}^{l} v^k$.

A few things are worth noting about Theorem 6.3.1. First, the theorem guarantees convergence for learning rates of the form $\alpha_k = \frac{C_\alpha}{\sqrt{k}}$ and $\beta_k = \frac{C_\beta}{\sqrt{k}}$, where the constants C_α, C_β can be *any* positive scalars. Even if the minimization learning rate is much larger than the maximization rate (or visa versa), the method is still asymptotically stable.

Also, we make a variety of smoothness assumptions about f and boundedness assumptions on the stochastic gradients, in addition to an assumption about the boundedness of the iterates. These assumptions are standard; indeed they are required to prove convergence of standard SGD methods for *weakly* convex (non-saddle) problems. Note that some of these assumptions could be dropped, and a faster rate could be proved, if we assume *strong* convexity.

6.4 Proof of Theorems

Assume the optimal solution $(u^\star, v^\star)$ exists, then $\mathcal{L}'_u(u^\star, v) = \mathcal{L}'_v(u, v^\star) = 0$. In the following proofs, we use $g_u(u, v)$, $g_v(u, v)$ to represent the stochastic approximation of gradients, where $\mathbb{E}[g_u(u, v)] = \mathcal{L}'_u(u, v)$, $\mathbb{E}[g_v(u, v)] = \mathcal{L}'_v(u, v)$. We show the convergence of the proposed stochastic primal-dual gradients for the primal-dual gap $P(u_k, v_k) = \mathcal{L}(u_k, v^\star) - \mathcal{L}(u^\star, v_k)$. We prove the $O(1/\sqrt{k})$ convergence rate in Theorem 6.3.1 by using Lemma 6.4.1 and Lemma 6.4.2, which present the contraction of primal and dual updates, respectively.

Lemma 6.4.1. *Suppose $\mathcal{L}(u, v)$ is convex in u and $\mathbb{E}[\|g_u(u, v)\|^2] \leq G_u^2$, we have*

$$\mathbb{E}[\mathcal{L}(u_k, v_k)] - \mathbb{E}[\mathcal{L}(u^\star, v_k)] \leq \frac{1}{2\alpha_k}\left(\mathbb{E}[\|u_k - u^\star\|^2] - \mathbb{E}[\|u_{k+1} - u^\star\|^2]\right) + \frac{\alpha_k}{2} G_u^2 \quad (6.9)$$

Proof. Use primal update in (1), we have

$$\|u_{k+1} - u^\star\|^2 = \|u_k - \alpha_k\, g_u(u_k, v_k) - u^\star\|^2 \tag{6.10}$$

$$= \|u_k - u^\star\|^2 - 2\alpha_k\, \langle g_u(u_k, v_k),\, u_k - u^\star\rangle + \alpha_k^2\, \|g_u(u_k, v_k)\|^2. \tag{6.11}$$

Take expectation on both side of the equation, substitute with $\mathbb{E}[g_u(u, v)] = \mathcal{L}'_u(u, v)$ and apply $\mathbb{E}[\|g_u^2(u, v)\|] \le G_u^2$ to get

$$\mathbb{E}[\|u_{k+1} - u^\star\|^2] \le \mathbb{E}[\|u_k - u^\star\|^2] - 2\alpha_k\, \mathbb{E}[\langle \mathcal{L}'_u(u_k, v_k),\, u_k - u^\star\rangle] + \alpha_k^2 G_u^2. \tag{6.12}$$

Since $\mathcal{L}(u, v)$ is convex in u, we have

$$\langle \mathcal{L}'_u(u_k, v_k),\, u_k - u^\star\rangle \ge \mathcal{L}(u_k, v_k) - \mathcal{L}(u^\star, v_k). \tag{6.13}$$

(6.9) is proved by combining (6.12) and (6.13). □

Lemma 6.4.2. *Suppose $\mathcal{L}(u, v)$ is concave in v and has Lipschitz gradients, $\|\mathcal{L}'_v(u_1, v) - \mathcal{L}'_v(u_2, v)\| \le L_v\|u_1 - u_2\|$; and bounded variance, $\mathbb{E}[\|g_u(u, v)\|^2] \le G_u^2$, $\mathbb{E}[\|g_v(u, v)\|^2] \le G_v^2$; and $\mathbb{E}[\|v_k - v^\star\|^2] \le D_v^2$, we have*

$$\begin{aligned}&\mathbb{E}[\mathcal{L}(u_k, v^\star)] - \mathbb{E}[\mathcal{L}(u_k, v_k)] \le \\ &\quad \frac{1}{2\beta_k}\left(\mathbb{E}[\|v_k - v^\star\|^2] - \mathbb{E}[\|v_{k+1} - v^\star\|^2]\right) + \frac{\beta_k}{2} G_v^2 + \alpha_k L_v\,(G_u^2 + D_v^2).\end{aligned} \tag{6.14}$$

Proof. From the dual update in (1), we have

$$\|v_{k+1} - v^\star\|^2 = \|v_k + \beta_k\, g_v(\bar{u}_{k+1}, v_k) - v^\star\|^2 \tag{6.15}$$

$$= \|v_k - v^\star\|^2 + 2\beta_k \,\langle g_v(\bar{u}_{k+1}, v_k),\, v_k - v^\star\rangle + \beta_k^2 \,\|g_v(\bar{u}_{k+1}, v_k)\|^2. \tag{6.16}$$

Take expectation on both sides of the equation, substitute $\mathbb{E}[g_v(u,v)] = \mathcal{L}'_v(u,v)$, and apply $\mathbb{E}[\|g_v^2(u,v)\|] \leq G_v^2$ to get

$$\mathbb{E}[\|v_{k+1} - v^\star\|^2] \leq \mathbb{E}[\|v_k - v^\star\|^2] + 2\beta_k\, \mathbb{E}[\langle \mathcal{L}'_v(\bar{u}_{k+1}, v_k),\, v_k - v^\star\rangle] + \beta_k^2\, G_v^2. \tag{6.17}$$

Reorganize (6.17) to get

$$\mathbb{E}[\|v_{k+1} - v^\star\|^2] - \mathbb{E}[\|v_k - v^\star\|^2] - \beta_k^2\, G_v^2 \leq 2\beta_k\, \mathbb{E}[\langle \mathcal{L}'_v(\bar{u}_{k+1}, v_k),\, v_k - v^\star\rangle]. \tag{6.18}$$

The right hand side of (6.18) can be represented as

$$\mathbb{E}[\langle \mathcal{L}'_v(\bar{u}_{k+1}, v_k),\, u_k - v^\star\rangle] \tag{6.19}$$

$$= \mathbb{E}[\langle \mathcal{L}'_v(\bar{u}_{k+1}, v_k) - \mathcal{L}'_v(u_k, v_k) + \mathcal{L}'_v(u_k, v_k),\, v_k - v^\star\rangle] \tag{6.20}$$

$$= \mathbb{E}[\langle \mathcal{L}'_v(\bar{u}_{k+1}, v_k) - \mathcal{L}'_v(u_k, v_k),\, v_k - v^\star\rangle] + \mathbb{E}[\langle \mathcal{L}'_v(u_k, v_k),\, v_k - v^\star\rangle], \tag{6.21}$$

where

$$\mathbb{E}[\langle \mathcal{L}'_v(\bar{u}_{k+1}, v_k) - \mathcal{L}'_v(u_k, v_k),\ v_k - v^\star \rangle] \tag{6.22}$$

$$\leq \mathbb{E}[\|\mathcal{L}'_v(\bar{u}_{k+1}, v_k) - \mathcal{L}'_v(u_k, v_k)\|\ \|v_k - v^\star\|] \tag{6.23}$$

$$\leq \mathbb{E}[L_v\ \|\bar{u}_{k+1} - u_k\|\ \|v_k - v^\star\|] \tag{6.24}$$

$$= \mathbb{E}[2L_y\ \|u_{k+1} - u_k\|\ \|v_k - v^\star\|] \tag{6.25}$$

$$= \mathbb{E}[2L_y\ \|\alpha_k g_u(u_k, v_k)\|\ \|v_k - v^\star\|] \tag{6.26}$$

$$\leq L_y \alpha_k\ \mathbb{E}[\ \|g_u(u_k, v_k)\|^2 + \|v_k - v^\star\|^2] \tag{6.27}$$

$$\leq L_y \alpha_k\ (G_u^2 + D_v^2). \tag{6.28}$$

Lipschitz smoothness is used for (6.24); the prediction step in (1) is used for (6.25); the primal update in (1) is used for (6.26); bounded assumptions are used for (6.28).

Since $\mathcal{L}(u, v)$ is concave in v, we have

$$\langle \mathcal{L}'_v(u_k, v_k),\ v_k - v^\star \rangle \leq \mathcal{L}(u_k, v_k) - \mathcal{L}(u_k, v^\star). \tag{6.29}$$

Combine equations (6.18, 6.21, 6.28) to get (6.29)

$$\begin{aligned} &\frac{1}{2\beta_k}\ \left(\mathbb{E}[\|v_{k+1} - v^\star\|^2] - \mathbb{E}[\|v_k - v^\star\|^2]\right) - \frac{\beta_k}{2} G_v^2 \\ &\leq L_v \alpha_k\ (G_u^2 + D_v^2) + \mathbb{E}[\mathcal{L}(u_k, v_k)] - \mathbb{E}[\mathcal{L}(u_k, v^\star)]. \end{aligned} \tag{6.30}$$

Rearrange the order of (6.30) to achieve (6.14). □

We now present the proof of Theorem 6.3.1.

Proof. Combining (6.9) and (6.14) in the Lemmas, the primal-dual gap $P(u_k, v_k) = \mathcal{L}(u_k, v^\star) - \mathcal{L}(u^\star, v_k)$ satisfies,

$$\begin{aligned}\mathbb{E}[P(u_k, v_k)] \leq & \frac{1}{2\alpha_k}\left(\mathbb{E}[\|u_k - u^\star\|^2] - \mathbb{E}[\|u_{k+1} - u^\star\|^2]\right) + \frac{\alpha_k}{2}G_u^2 \\ & + \frac{1}{2\beta_k}\left(\mathbb{E}[\|v_k - v^\star\|^2] - \mathbb{E}[\|v_{k+1} - v^\star\|^2]\right) + \frac{\beta_k}{2}G_v^2 + \alpha_k L_v\left(G_u^2 + D_v^2\right).\end{aligned} \tag{6.31}$$

Accumulate (6.31) from $k = 1, \ldots, l$ to obtain

$$\begin{aligned}&\sum_{k=1}^{l}\mathbb{E}[P(u_k, v_k)] \leq \\ &\frac{1}{2\alpha_1}\mathbb{E}[\|u^1 - u^\star\|^2] + \sum_{k=2}^{l}(\frac{1}{2\alpha_k} - \frac{1}{2\alpha_{k-1}})\mathbb{E}[\|u_k - u^\star\|^2] + \sum_{k=1}^{l}\alpha_k(\frac{G_u^2}{2} + L_v G_u^2 + L_v D_v^2) \\ &+ \frac{1}{2\beta_1}\mathbb{E}[\|v^1 - v^\star\|^2] + \sum_{k=2}^{l}(\frac{1}{2\beta_k} - \frac{1}{2\beta_{k-1}})\mathbb{E}[\|v_k - v^\star\|^2] + \sum_{k=1}^{l}\beta_k\frac{G_v^2}{2}.\end{aligned} \tag{6.32}$$

Assume $\mathbb{E}[||u_k - u^\star\|^2] \leq D_u^2$, $\mathbb{E}[||v_k - v^\star\|^2] \leq D_v^2$ are bounded, we have

$$\begin{aligned}\sum_{k=1}^{l}\mathbb{E}[P(u_k, v_k)] \leq & \frac{1}{2\alpha_1}D_u^2 + \sum_{k=2}^{l}(\frac{1}{2\alpha_k} - \frac{1}{2\alpha_{k-1}})D_u^2 + \sum_{k=1}^{l}\alpha_k(\frac{G_u^2}{2} + L_v G_u^2 + L_v D_v^2) \\ & + \frac{1}{2\beta_1}D_v^2 + \sum_{k=2}^{l}(\frac{1}{2\beta_k} - \frac{1}{2\beta_{k-1}})D_v^2 + \sum_{k=1}^{l}\beta_k\frac{G_v^2}{2}.\end{aligned} \tag{6.33}$$

Since α_k, β_k are decreasing and $\sum_{k=1}^{l}\alpha_k \leq C_\alpha\sqrt{l+1}$, $\sum_{k=1}^{l}\beta_k \leq C_\beta\sqrt{l+1}$,

we have

$$\sum_{k=1}^{l}\mathbb{E}[P(u_k,v_k)]\leq\frac{\sqrt{l}}{2}\left(\frac{D_u^2}{C_\alpha}+\frac{D_v^2}{C_\beta}\right)+\sqrt{l+1}\left(\frac{C_\alpha G_u^2}{2}+C_\beta L_v G_u^2+C_\alpha L_v D_v^2+\frac{C_\beta G_v^2}{2}\right) \tag{6.34}$$

For $\hat{u}^l=\frac{1}{l}\sum_{k=1}^{l}u_k$, $\hat{v}^l=\frac{1}{l}\sum_{k=1}^{l}v_k$, because $\mathcal{L}(u,v)$ is convex-concave, we have

$$\begin{aligned}\mathbb{E}[P(\hat{u}^l,\hat{v}^l)]&=\mathbb{E}[\mathcal{L}(\hat{u}^l,v^\star)-\mathcal{L}(u^\star,\hat{v}^l)] &(6.35)\\ &\leq\mathbb{E}[\frac{1}{l}\sum_{k=1}^{l}(\mathcal{L}(u_k,v^\star)-\mathcal{L}(u^\star,v_k))] &(6.36)\\ &=\frac{1}{l}\sum_{k=1}^{l}\mathbb{E}[\mathcal{L}(u_k,v^\star)-\mathcal{L}(u^\star,v_k)] &(6.37)\\ &=\frac{1}{l}\sum_{k=1}^{l}\mathbb{E}[P(u_k,v_k)]. &(6.38)\end{aligned}$$

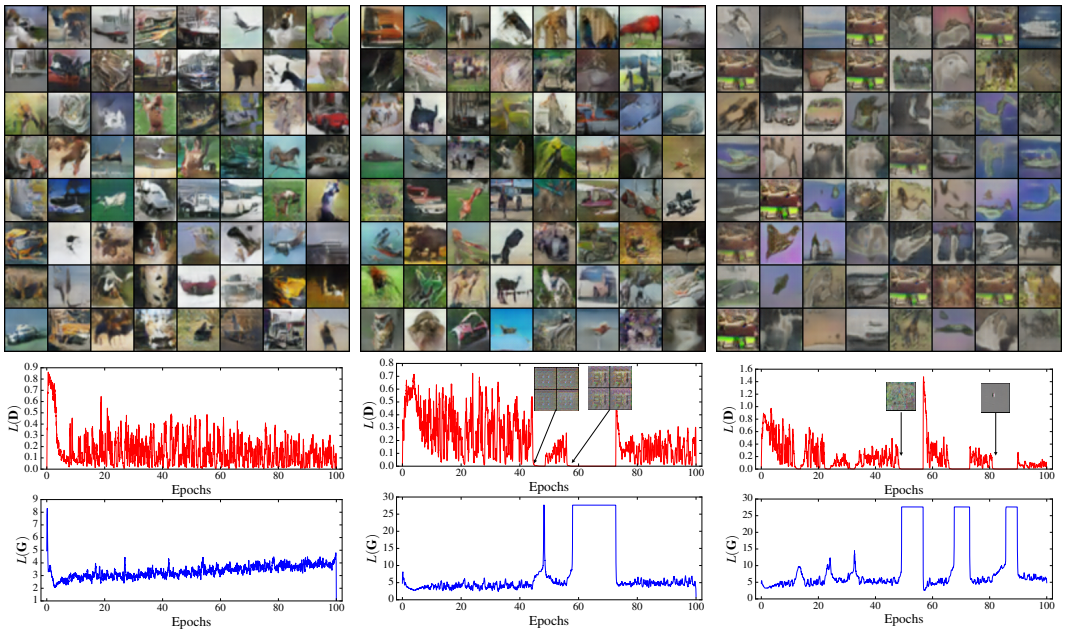
Combine (6.34) and (6.38) to prove

$$\mathbb{E}[P(\hat{x}^l,\hat{y}^l)]\leq\frac{1}{2\sqrt{l}}\left(\frac{D_u^2}{C_\alpha}+\frac{D_v^2}{C_\beta}\right)+\frac{\sqrt{l+1}}{l}\left(\frac{C_\alpha G_u^2}{2}+C_\alpha L_v G_u^2+C_\alpha L_v D_v^2+\frac{C_\beta G_v^2}{2}\right). \tag{6.39}$$

□

6.5 Generative Adversarial Network

We apply stochastic alternating method with prediction step to stabalize the training of GAN, as shown in Fig. 6.1. We refer readers to [Yad+18] for more details on our GAN experiments.



(a) With **G** prediction (b) DCGAN [Rad+16] (c) Unrolled GAN [Met+16]

Figure 6.1: Comparison of GAN training algorithms for DCGAN architecture on Cifar-10 image datasets. Using default parameters of DCGAN; $lr = 0.0002, \beta_1 = 0.5$.

Chapter 7: Adversarial Network for Image Processing

In this chapter, we use GAN framework to boost the performance of image processing tasks: image style transfer and image dehazing. In Section 7.1, we apply adversarial network to learn from multi-domain artistic images for arbitrary style transfer. In Section 7.2, we introduce a simple yet effective network as a strong baseline for singe image dehazing. We show that the performance of deep network can be improved with the help of normalization layers and pre-trained encoder. We also use GAN framework of image dehazing so that we can train network without using paired clean and hazy images that are difficult to acquire. The encoder-decoder architecture and fully convolutional network in this chapter can also be used for other tasks such as semantic segmentation [Hsu+18]. This chapter is based on our work presented in [Xu+18a; Xu+19a; Yan+18]

7.1 Image Style Transfer

7.1.1 Introduction

Image style transfer is a task that aims to render the content of one image with the style of another, which is important and interesting for both practical

and scientific reasons. The style transfer techniques can be widely used in image processing applications such as mobile camera filters and artistic image generation. Furthermore, the study of style transfer often reveals the intrinsic property of images. Style transfer is challenging as it is difficult to explicitly separate and represent the content and style of an image.

In the seminal work of Gatys et al. [Gat+16], the authors represent content with deep features extracted by a pre-trained neural network, and represent style with second order statistics (i.e. the Gram matrix) of the deep features. They propose an optimization framework with the objective that the generated image has similar deep features to the given content image, and similar second order statistics to the given style image. The generated results are visually impressive, but the optimization framework is far too slow for real-time applications. Later works [Joh+16; Uly+17b] train a feed-forward network to replace the optimization framework for fast stylization, with a loss similar to Gatys et al. [Gat+16]. However, they need to train a network for each style image and cannot generalize to unseen images. More recent approaches [HB17; Li+17f] tackle arbitrary style transfer for unseen content and style images, which still represent style with second order statistics of deep features. The second order statistics of style representation is originally designed for *textures* [Gat+15], and style transfer is considered as texture transfer in previous methods.

Another line of research considers style transfer as conditional image generation, and apply adversarial networks to train an image to image translation network [Iso+17; Tai+17; Zhu+17b; Hua+18]. The trained image translation networks can

transfer image from one domain to another domain, for example, from a natural image to sketch. However, they cannot be applied to arbitrary style transfer as the input images are from mutliple domains.

In this paper, we combine the best of both worlds by adversarially training a single feed-forward network for arbitrary style transfer. We introduce several techniques to tackle the challenging problem of adversarial training from multi-domain data. In adversarial training, the generator (stylization network) and the discriminator are alternatively updated. Both our generator and discriminator are conditional networks. The generator is trained to fool the discriminator, as well as satisfy the content and style representation similarity to inputs. Our generator is built upon a state-of-the-art network for abitrary style transfer [HB17], which is conditioned on both content image and style image, and uses adaptive instance normalization (AdaIN) to combine the two inputs. AdaIN shifts the mean and variance of the deep features of content image to match those of the style image. Our discriminator is conditioned on the coarse domain categories, which is trained to distinguish the generated images with real images from the the same style category.

Comparing with previous arbitrary style transfer methods, our approach uses the discriminator to learn a data-driven representation for styles. The combined loss for our generator considers both instance-level information from style loss and category-level information from adversarial training. Comparing with previous adversarial training methods, our approach handles multi-domain inputs by using a conditional generator designed for arbitrary style transfer and a conditional discriminator. Moreover, we propose a mask module to automatically control the level

of stylization by predicting a mask to blend the stylized features and the content features. Finally, we use the trained discriminator to rank and find the representative generated images in each style category. We release our code and model at `https://github.com/nightldj/behance_release`

7.1.2 Related work

Style transfer. We briefly review the neural style transfer methods, and recommend [Jin+17] for a more comprehensive review. Gatys et al. [Gat+16] proposed the first neural style transfer method based on an optimization framework, which uses deep features to represent content and Gram matrix to represent style. The optimization framework was replaced by a feed forward network to achieve real-time performance in [Joh+16; Uly+16b; Wan+17]. Ulyanov et al. [Uly+17b] showed that instance normalization is particularly effective for training a fast style transfer network. Other works focused on controlling spatial, color, and stroke for stylization [Gat+17; Fri+16; Jin+18], and exploring other style representation such as mean and variance [Li+17e], histogram [Wil+17b], patch-based MRF [LW16a], and patch-based GAN [LW16b]. Comparing with [Gat+16], these fast style transfer methods sometimes compromise on the visual quality, and need to train one network for each style.

Various methods have been proposed to train a single feed forward network for multiple styles. Dumoulin et al. [Dum+17] proposed conditional instance normalization, which learned the affine parameter for each style image. Chen et al.

[Che+17a] learned the "style bank", which contains several layers of filters for each style. Zhang and Dana [ZD17] proposed comatch layers for multi-style transfer. These methods only work with limited number of styles, and cannot apply to an unseen style image.

More recent approaches are designed for arbitrary style transfer, where both the content and the style inputs can be unseen images. Ghiasi et al. [Ghi+17] extended conditional instance normalization (IN) by training a separate network to predict the affine parameter of IN. Falong Shen and Zeng [FSZ18] learned a meta network to predict filters in the transformation networks. Huang and Belongie [HB17] proposed adaptive instance normalization (AdaIN) that adjusts the mean and variance of content image to match those of the style image. Li et al. [Li+17f; Li+18c] used feature whitening and coloring transforms (WCT) to match the statistics of the content image to the style image. Sheng et al. [She+18] proposed feature decoration that generalizes AdaIN and WCT. Note that the optimization framework [Gat+16] and path-based non-parametric methods (e.g., style swamp [CS16], deep image analogy[Lia+17], and deep feature reshuffle [Gu+18]) can also be applied to arbitrary style transfer, but these methods can be much slower. Zhang et al. [Zha+18b] proposed to separate style and content and then combine them with bilinear layer, which requires a set of content and style images as input and has limited applications. Our approach is the first to explore adversarial training for arbitrary style transfer.

Generative adversarial networks (GANs). GANs have been widely studied for image generation and manipulation tasks since [Goo+14a]. Elgammal et al.

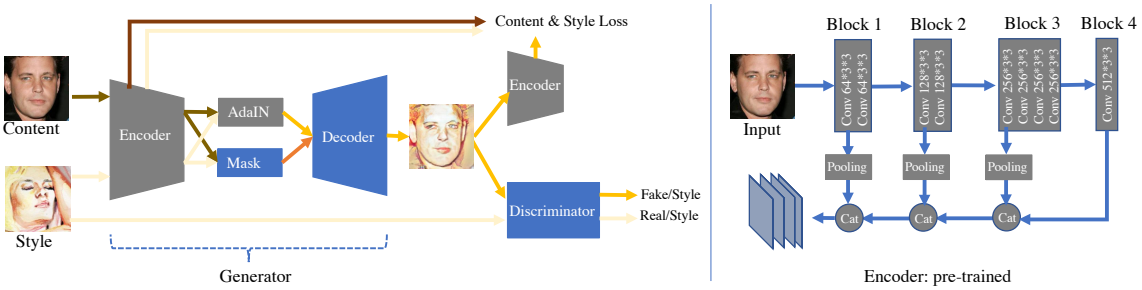


Figure 7.1: Proposed network: (left) encoder-decoder as generator; (right) pre-trained VGG as encoder. The decoder architecture is symmetric comparing to encoder. We use the conventional texture loss based on pre-trained encoder features, and adversarially train mask module, decoder and discriminator.

[Elg+17] applied GANs to generate artistic images. Isola et al. [Iso+17] used conditional adversarial networks to learn the loss for image to image translation, which is extended by several concurrent methods [Zhu+17b; Kim+17; Yi+17; Liu+17] that explored cycle-consistent loss when training data are unpaired. Later works improved the diversity of generated images by considering multimodality of data [Zhu+17a; Alm+18; Hua+18]. Similar techniques have been applied to specific image to image translation tasks such as image dehazing [Yan+18], face to cartoon [Tai+17; Roy+17] and font style transfer [Aza+18]. These methods successfully train a transformation network from one image domain to another. However, they cannot handle multi-domain input and output images, and it is known to be difficult to generate images with large variance [Che+16c; Ode+17; MK18]. Our approach adopt conditional generator and discriminator to tackle the multi-domain input and output for arbitrary style transfer.

7.1.3 Proposed method

We use an encoder-decoder architecture as our transformation network, and use the convolutional layers of the pre-trained VGG net [SZ14; Xu+18a] as our encoder to extract the deep features. We add skip connections and concatenate the features from different levels of convolutional layers as the output feature of the encoder. We adopt adaptive instance normalization (AdaIN) [HB17] to adjust the first and second order statistics of the deep features. Furthermore, we generate spatial masks to automatically adjust the stylization level. Our transformation network is a conditional generator inspired by the state-of-the-art network for arbitrary style transfer. Our network is trained with perceptual loss for content representation, Gram loss for style representation as in [Gat+16; Joh+16; Uly+16b], as well as the adversarial loss to capture the common style information beyond textures from a style category. We show the proposed network in figure 7.1, and provide details in the following sections.

7.1.3.1 Network architecture

Our **encoder** uses the convolutional layers of the VGG net [SZ14] pre-trained on Imagenet large-scale image classification task [Rus+15]. VGG net contains five blocks of convolutional layers, and we adopt the first three blocks and the first convolutional layer of the forth block. Each block contains convolutional layers with ReLU activation [Kri+12], and the width (number of channels) and size (height and width) of the convolutional layers are shown in figure 7.1. There is a maxpooling

layer of stride two between blocks, and the width of convolutional layer is doubled after the downsampling by maxpooling. We concatenate the features from the first convolutional layer of each block as the output of the encoder. These skip connections help to transfer style captured by both high-level and low-level features, as well as make the training easier by smoothing the loss surface of neural networks [Li+18b].

Our **decoder** is designed to be almost symmetric to the VGG encoder, which has four blocks and between blocks are transposed convolutional layer for upsampling. We add LeakyReLU [He+15] and batch normalization [IS15] to each convolutional layer for effective adversarial training [Rad+16]. The decoder is trained from scratch.

Adaptive instance normalization (AdaIN) has been shown to be effective for image style transfer [HB17]. AdaIN shifts the mean and variance of deep features of content to match style with no learnable parameters. Let $x, y \in \mathbb{R}^{N \times C \times H \times W}$ represent the features of a convolutional layer from a minibatch of content and style images, where N is the batch size, C is the width of the layer (number of channels), H and W are height and width, respectively. x_{nchw} denotes the element at height h, width w of the cth channel from the nth sample, and adaIN layer can be written as,

$$A_{nchw}(x, y) = \sigma_{nc}(y) \left(\frac{x_{nchw} - \mu_{nc}(x)}{\sigma_{nc}(x)} \right) + \mu_{nc}(y) \tag{7.1}$$

where $\mu_{nc}(x) = {}^{1}/_{HW} \sum_{h,w=1}^{H,W} x_{nchw}$, $\sigma_{nc}(x) = \sqrt{{}^{1}/_{HW} \sum_{h,w=1}^{H,W} (x_{nchw} - \mu_{nc})^2 + \epsilon}$, ϵ is a very small constant, and $\mu_{nc}(x), \sigma_{nc}^2(x)$ represent the mean and variance for the

cth channel of the nth sample of feature x.

The **mask** module in our network contains a few convolutional layers operated on the concatenation of content feature x and style feature y. The output is a spatial soft mask $M(x, y) \in [-1, 1]^{N \times C \times H \times W}$ that has the same size as feature and each value is between -1 and 1. The generated mask $M(x, y)$ is used to control the stylization level by linearly combine the adaIN feature $A(x, y)$ and the original content feature s as the input of the decoder,

$$z = M(x, y) \times x + (1 - M(x, y)) \times A(x, y), \tag{7.2}$$

where the element-wise operations are used for combining these features.

Our **discriminator** is a patch-based network inspired by [Iso+17]. To handle the multi-domain images for arbitrary style transfer, our discriminator is conditioned on the style category labels. Inspired by AC-GAN [Ode+17], our discriminator predicts the style category and distinguish the real image and fake image at the same time. We also adopt the projection discriminator [MK18] to make sure the style category conditioning will not be ignored.

7.1.3.2 Adversarial training

We alternatively update the generator (mask module and decoder) and discriminator during training, and apply prediction optimizer [Yad+18] to stabilize the training.

Generator update. Our generator takes a content image and a style image

as input, and outputs the stylized image. The generator is updated by minimizing the loss combined of adversarial loss $\mathcal{L}_A$, style classification loss $\mathcal{L}_{DS}$, content loss $\mathcal{L}_c$ and style loss $\mathcal{L}_s$,

$$\min_G \mathcal{L}_G = \mathcal{L}_A + \lambda_{DS}\mathcal{L}_{DS} + \lambda_c\mathcal{L}_c + \lambda_s\mathcal{L}_s, \tag{7.3}$$

where $\lambda_{DS}, \lambda_c, \lambda_s$ are hyperparameters for the weights of different losses. Let us denote the feature map of the lth layer in our encoder as $x^{(l)}, y^{(l)}$, the input content and style images as $x^{(0)}, y^{(0)}$, the generator network as $G(\cdot,\cdot)$, and the discriminator network as $D(\cdot)$. When the discriminator $D(\cdot)$ is fixed, the output stylized images $\hat{x} = G(x^{(0)}, y^{(0)})$ aim to fool the discriminator, and also be classified to same style category s as the input style image,

$$\begin{aligned} \mathcal{L}_A &= \mathbb{E}[\log \text{Prob}(\text{Real}|D(\hat{x}))], \\ \mathcal{L}_{DS} &= \mathbb{E}[\log \text{Prob}(s|D(\hat{x}))]. \end{aligned} \tag{7.4}$$

$\mathcal{L}_A$ and $\mathcal{L}_{DS}$ are learned loss that capture the category-level style of images from the training data. We also use the traditional content and style loss based on deep features and Gram matrix,

$$\begin{aligned} \mathcal{L}_c &= \mathbb{E}[\|x^{(4)} - \hat{x}^{(4)}\|_1], \\ \mathcal{L}_s &= \mathbb{E}[\sum_{l=1}^{4} \|\text{Gram}(y^{(l)}) - \text{Gram}(\hat{x}^{(l)})\|_1]. \end{aligned} \tag{7.5}$$

We use the deep feature from the forth block of pre-trained VGG net for content

Figure 7.2: Benefits of adversarial training and mask module. We show the encoder-decoder network with adversarial training only, mask module only, and the combination of adversarial training and mask module. Mask module only does not improve the visual quality of generated images, which have artifacts and undesired textures. GAN only can generate collapsed images with corrupted eyes and noses.

representation, and use the Gram matrix from all the blocks for style representation. We find ℓ_1 norm is more stable than ℓ_2 when combining with the adversarial loss.

Discriminator update. Our discrimintor is conditioned on style category to handle the multi-domain generated images, inspired by [Che+16c; Ode+17; MK18; Xu+18b]. When the generator is fixed, the discriminator is adversarially trained to distinguish the generated images and the real style images,

$$\min_D \mathcal{L}_D = \hat{\mathcal{L}}_A + \lambda_{DS}\hat{\mathcal{L}}_{DS}, \tag{7.6}$$

where $\hat{\mathcal{L}}_A = \mathbb{E}[\log \text{Prob}(\text{Fake}|D(\hat{x})) + \log \text{Prob}(\text{Real}|D(y))]$, and $\hat{\mathcal{L}}_{DS} = \mathbb{E}[\log \text{Prob}(s|D(\hat{x})) + \log \text{Prob}(s|D(y))]$.

Discriminator for ranking. The adversarilly trained discriminator characterizes the real style images, and hence can be used to rank the generated im-

ages. We rank the stylized images $\hat{x}$ based on the likelihood score $\text{Prob}(s|D(\hat{x})) * \text{Prob}(\text{Real}|D(\hat{x}))$.

7.1.3.3 Ablation study

The encoder-decoder architecture and adaIN module have been shown to be effective in previous work [HB17]. We use visual examples to show the importance of mask module and adversarial training in the proposed method in figure 7.2. We present results from adversarially trained network without mask module, network with mask module but trained without adversarial loss, and the proposed method. When trained without adversarial loss, the network produces visually similar results with or without mask module as the network is over-parameterized.

	vectorart	3D graphics	comic	graphite	oil paint	pen ink	water color	all
AdaIN [HB17]	0.2849	0.2029	0.2314	0.1277	0.3018	0.2151	0.2118	0.2199
WCT [Li+17f]	0.1134	0.1957	0.2066	**0.4754**	0.3350	0.2868	**0.4409**	0.3001
Ours	**0.6017**	**0.6014**	**0.5620**	0.3969	**0.3632**	**0.4981**	0.3473	**0.4800**

Table 7.1: Quantitative evaluation for style transfer. Our method is preferred by human annotators and outperforms baselines.

Our adversarial training significantly improves the visual quality of the generated images in general. The block effects and many other artifacts are removed through adversarial training, which makes the generated images look more "natural". Moreover, the data-driven discriminator learns to distinguish foreground and background well; adversarial training cleans the background and adds more details to the foreground. Our mask module controls the stylization level at different spatial location of the image, which significantly improves the stylization of salient

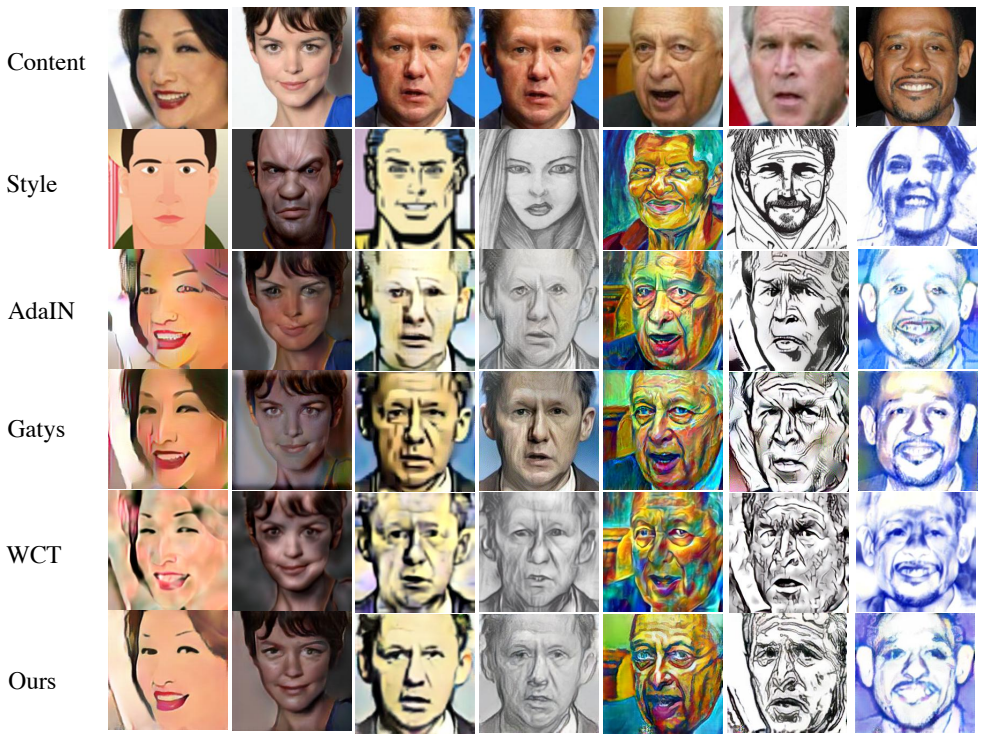


Figure 7.3: Qualitative evaluation for style transfer. We shown examples of transferring photos to seven different styles. AdaIN and WCT will generate artifacts and undesired textures. Gatys' results are more visually appealing, but the optimization is slow, and it is hard to choose the parameter to control stylization level. Our method efficiently generate clean and stylized images.

components like eyes, nose and mouth of a face. The salient regions are repeatedly captured by the deep features from high-level layers, which can make them difficult to handle when adjusting the statistics of the features. By controlling the stylization level, the mask module prevents over-stylization of salient region, and also helps adversarial training by relieving the mode collapse of salient regions.

7.1.4 Experiments

We qualitatively and quantitatively evaluate the proposed method with experiments. We extensively use the Behance dataset [Wil+17a] for training and testing. Behance [Wil+17a] is a large-scale dataset of artistic images, which contains coarse

category labels for content and style. We use the seven media labels in Behance as style category: vector art, 3D graphics, comic, graphite , oil paint, pen ink, and water color. We create four subsets from the Behance images for face, bird, car, and building. Our face dataset is created by running a face detector on a subset of images with people as content label and contains roughly 15,000 images for each style. The other three are created by selecting the top 5000 ranked images of each media for the content, respectively. We add describable textures Dataset (DTD) [Cim+14] as another style category to improve the robustness of our method. We add natural images as both content images and an extra style for each subset. Specifically, we use labeled faces in the wild (LFW) [Hua+07], the first 16,000 images of CelebA dataset [Liu+15], Caltech-UCSD birds dataset [Wel+10], cars dataset [Kra+13], and Oxford building dataset [Phi+07]. In total, we have nine style categories in our data. We split both content and style images into training/testing set, and use unseen testing images for our evaluation. The total number of training/testing images are 122,247 / 11,325 for face, 35,000 / 3,505 for bird, 36,940 / 3,700 for car, and 34,374 / 3,444 for building.

We train the network on face images, and then fine-tune it on bird, car, and building. We use Adam optimizer with prediction method [Yad+18] with learning rate $2e-4$ and parameter $\beta_1 = 0.5, \beta_2 = 0.9$. We train the network with batch size 56 for 150 epochs and linearly decrease the learning rate after 60 epochs. It takes about 8 hours to complete on a workstation with 4 GPUs. We set all weights in our combined loss (7.3) as 1 except for $\lambda_s = 200$ for the style loss. The weights are chosen so that different components of the loss have similar numerical scales. The

training code and pre-trained model in Pytorch are released in `https://github.com/nightldj/behance_release`.

We compare with arbitrary style transfer methods, the optimization framework of neural style transfer (Gatys) [Gat+16], and two state-of-the-art methods, adaptive instance normalization (AdaIN) [HB17] and feature transformation (WCT) [Li+17f]. Note that our approach, AdaIN and WCT apply feed-forward network for style transfer, which are much faster than Gatys method.

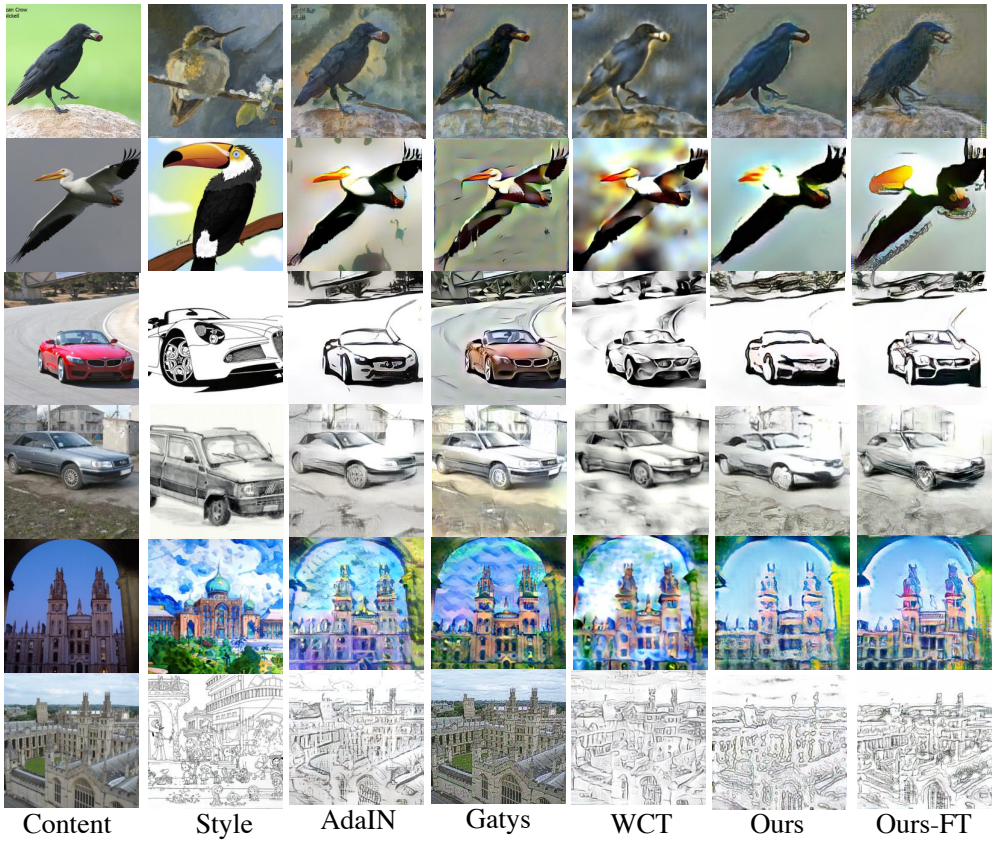


Figure 7.4: Qualitative evaluation for general objects. This task is more difficult for our GAN-based method because the training data is more noisy, especially for bird images with large diversity. Our method can generate clean background, detailed foreground, and better stylized strokes.

	vectorart	3D graphics	comic	graphite	oil paint	pen ink	water color	all
AdaIN [HB17]	0.2119	0.2703	0.3089	0.3260	0.2778	**0.3944**	0.3654	0.3203
WCT [Li+17f]	**0.4503**	**0.4865**	**0.3740**	0.1547	**0.4383**	0.2310	0.1731	0.3145
Ours	0.3377	0.2432	0.3171	**0.5193**	0.2840	0.3746	**0.4615**	**0.3652**

Table 7.2: Quantitative evaluation for style transfer of building. Different methods are competitive for different styles. The overall performance of our method is better.

Figure 7.5: Qualitative evaluation for style ranking.

7.1.4.1 Evaluation of style transfer

We qualitatively compare our approach with previous arbitrary style transfer methods, and present some results in figure 7.3. We show seven pairs of content and style images from our face dataset, and the style images are from testing set of vector art, 3D graphics, comic, graphite , oil paint, pen ink, and water color, respectively. For Gatys method [Gat+16], we tune the weight parameter, and select the best visual results from either Adam or BFGS as optimizer. For AdaIN [HB17] and WCT [Li+17f], we use their released best models. The content and style images are from the separate testing set that have not been seen for our approach and the baseline methods.

Gatys method [Gat+16] is sensitive to parameter and optimizer setting. We

may get results that are not stylized enough even after parameter tuning due to the difficulty of optimization. AdaIN [HB17] often over-stylizes the content image, creates undesirable artifacts, and sometimes changes the semantic of the content image. WCT [Li+17f] suffers from severe block effect and artifacts. The previous methods all create texture-like artifacts because of the texture-based style representation. For example, the stylized images of baselines in the first column of figure 7.3 have stride artifacts. Our approach generate more visually appealing results with clean background, vivid foreground, and more consistent with the style of the input.

We conduct user study on Amazon Mechanical Turk, and present quantitative results in table 7.1. We compare with the two recent fast style transfer methods in this study. We randomly select 10 content images and 10 style images from each Behance style category to generate 700 testing pairs. For each pair, we show the stylized images by our approach, AdaIN [HB17], and WCT [Li+17f], and ask 10 users to select the best results. We remove the unreliable results that are labeled too soon, and show preference (click) ratio for different style categories. WCT [Li+17f] performs well on graphite and water color, where the style images themselves are visually not "clean". Our approach achieves the best results in the other five categories and is overall the most favorable.

7.1.4.2 Evaluation of style transfer for general objects

We evaluate the performance of the proposed approach on general objects beyond face. Specifically, we test for bird, car, and building. In figure 7.4, we show

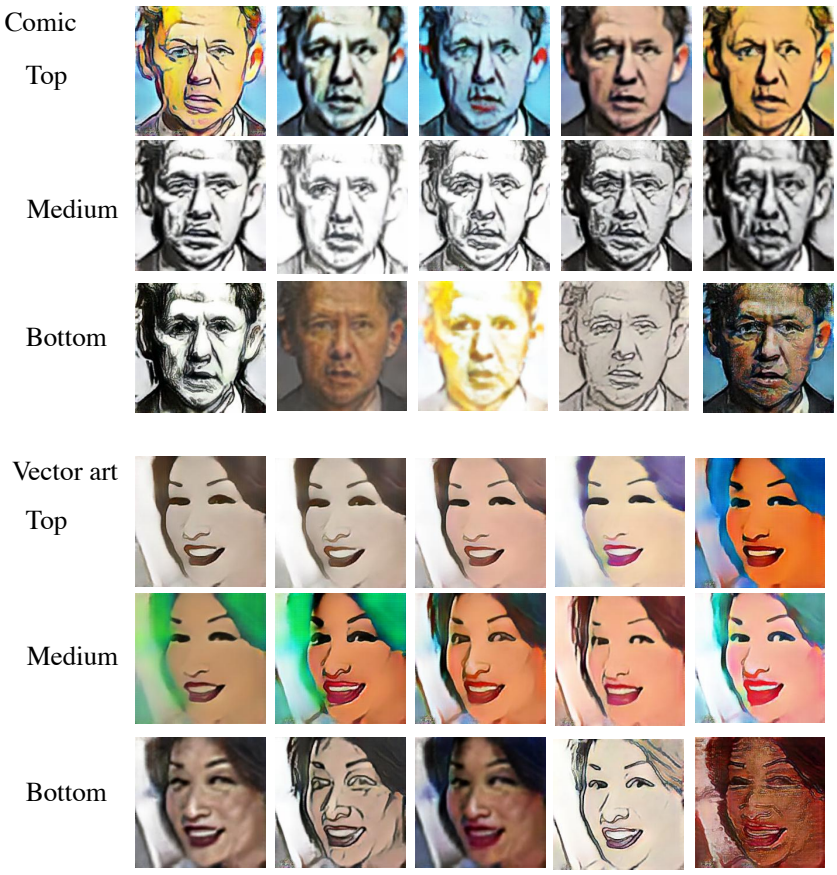


Figure 7.6: Ranking stylized images by our discriminator.

the stylized images generated by our network trained on face (Ours), as well as fine-tuned for each object (Outs-FT). Our network trained on face generalizes well, and generates images look comparable, if not better than, the baseline methods. Fine-tuning on bird does not help the performance. The adversarial training may be too difficult for bird because the given training style images are noisy and diverse. Fine-tuning on car and building brings more details to the foreground object of our generated images. The training images of car and building are also noisy and diverse, but these objects are more structured than bird. We show more results on our performance on general object tasks in the supplementary material.

We conduct the user study for building images and report results in table 7.2. Our approach achieves good results for graphite and water color because of the clean background in our generated images. For the other categories, our results are comparable with baselines. Our overall performance is still the best.

7.1.4.3 Evaluation for style ranking

We apply the trained discriminator to rank the generated images for a style category. Figure 7.5 show the top five generated images by stylizing with all the testing images in comic style. The stylized images are generated by our network, and ranked by our discriminator, a style classifier, and random selection, respectively. The style classifier use the same network architecture as our discriminator and training data as our method. The hyper parameters are tuned to achieve the best style classification accuracy on the separate validation dataset, which makes

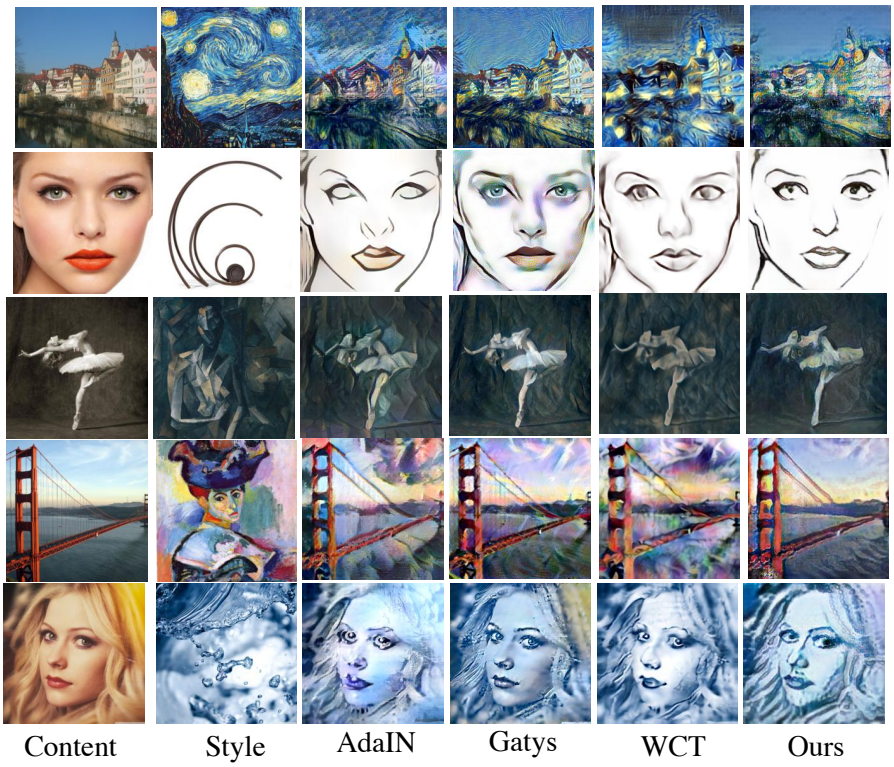


Figure 7.7: Qualitative evaluation for style transfer on texture-centric cases in previous papers. Our method generates stylized images with clean background, which are visually competitive to the previous methods that targeted only on texture transfer.

the style classifier a strong baseline. Our generator network produced good results, and even random selected images look acceptable. The top selected results of our discriminator are more diverse, and more consistent to the comic style because of the adversarial training.

Figure 7.6 shows more ranked images by our discriminator at top, in the middle, and at the bottom for two content images stylized by images from two categories. The top ranked results are more visually appealing, and more consistent with the style category.

Finally, we conduct user study to compare the ranking performance of our discriminator and the baseline classifier. We generated images by stylizing ten content

images with all the testing images for each of the seven Behance styles, and rank the 70 sets of results. We comparing the rank of each generated image by discriminator and classifier, and select five images that are ranked higher by our discriminator, and five images that are ranked higher by the baseline classifier. We show the ten images to ten users and ask them to select five images for each set. The preference ratio of our discriminator is 0.5068 comparing to 0.4932 of classifier. We beat a strong baseline in a highly subjective and challenging evaluation.

7.1.5 Supplemental experiments

In this section, we present supplemental experiments to show interesting side effects of the proposed method. We first demonstrate our method can be applied to previous style transfer test cases which focus on transferring textures of the style image. We then show that the proposed method can be applied to destylization and generate images look more realistic than baselines.

7.1.5.1 Examples for general style transfer

In figure 7.7, we evaluate on test cases from previous style transfer papers. The style images have rich texture information, and the content images vary from face to building. Our network is trained on our face dataset described in section 7.1.4. Our network generalizes well and produces comparable results, if not better than, comparing with baselines. Particularly, our approach often generates clean background without undesired artifacts.

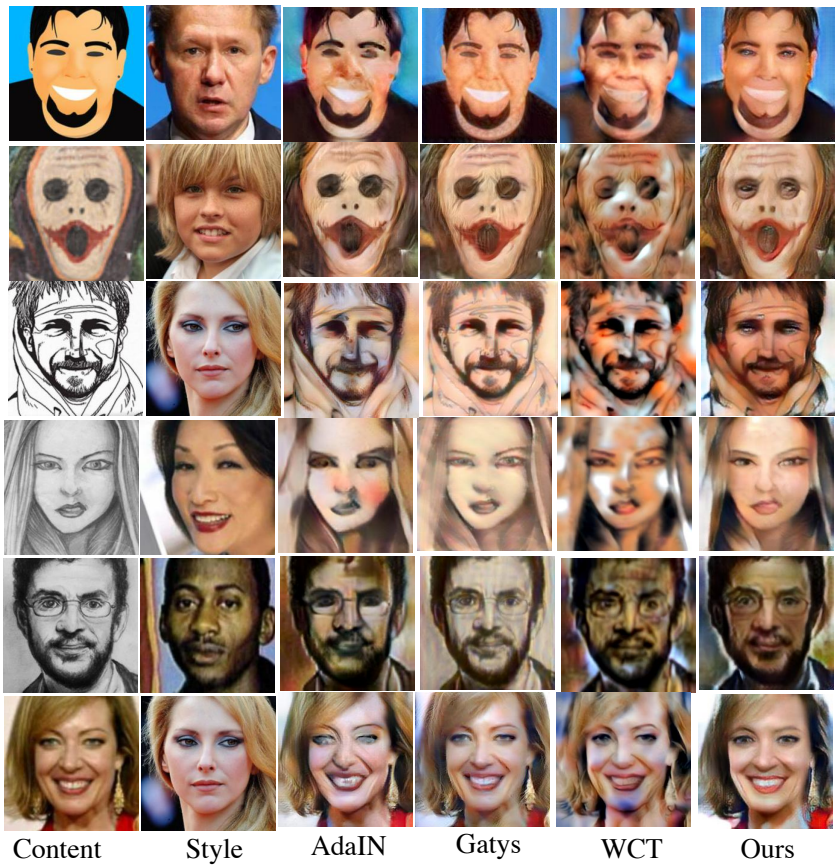


Figure 7.8: Qualitative evaluation for destylization.

7.1.5.2 Destylization

We show that if we also use artistic images as content images during training, the exact same architecture can be used to destylize images (figure 7.8). Destylization is a difficult task because we only use one network to destylize diverse artistic images. The training also becomes much more difficult as the number of pairs increase square to the samples. Though there is still room to improve, our adversarial training and network architecture look promising in limited training time. The last row in 7.8 also suggests our network can transfer style of photorealistic images, which is difficult for the baselines.

7.1.6 Summarization and discussion

We propose a feed-forward network that uses adversarial training to enhance the performance of arbitrary style transfer. We use both conditional generator and conditional discriminator to tackle multi-domain input and output. Our generator is inspired by the recent progress in arbitrary style transfer, and our discriminator is inspired by the recent progress in generative adversarial networks. Our approach combines the best of both worlds. We propose a mask module that helps in both adversarial training and style transfer. Moreover, we show that our trained discriminator can be used to select representative stylized image, which has been a long-standing problem.

Previous style transfer and GAN-based image translation methods only target on one domain, such as transferring the style of oil paint, or transforming from natural images to sketches. We systematically study the style transfer problem on a large-scale dataset of diverse artistic images. We can train one network to generate images in different styles, such as comic, graphite, oil paint, water color and vector art. Our approach generates more visually appealing results than previous style transfer methods, but there is still room to improve. For example, transferring image into 3D graphics with the arbitrary style transfer network is still challenging.

7.2 Image Dehazing

7.2.1 Introduction

Images captured in the wild are often degraded in visibility, colors, and contrasts caused by haze, fog and smoke. Recovering high-quality clear images from degraded images (a.k.a. image dehazing) is beneficial for both low-level image processing and high-level computer vision tasks. Dehazed images are more visually appealing to generate for image processing tasks. Dehazed images can improve the robustness of vision systems that often assume clear images as input. Typical applications that benefit from image dehazing include image super-resolution, visual surveillance, and autonomous driving. Image dehazing is highly desired because of the increasing demand of deploying visual system for real-world applications.

Image dehazing is a challenging problem. The effect of haze is caused by atmospheric absorption and scattering that depend on the distance of the scene points from the camera. In computer vision, the hazy image is often described by a simplified physical model, i.e., the atmospheric scattering model [McC76; NN02; He+11; Li+17b],

$$I(x) = J(x)t(x) + A(1 - t(x)), \tag{7.7}$$

where $I(x)$ is the observed hazy image, $J(x)$ is the scene radiance (clear image), $t(x)$ is the medium transmission map, and A is the global atmospheric light. When the atmosphere is homogeneous, $t(x)$ can be further expressed as a function of the scene depth d(x) and the scattering coefficient β of the atmosphere as $t(x) = \exp(-\beta d(x))$.

The goal of image dehazing is to recover clear image $J(x)$ from hazy image $I(x)$. Single image dehazing is particularly challenging. It is under-constrained because haze is dependent on many factors, including the unknown depth information that is difficult to recover from a single image.

The atmospheric scattering model (7.7) has been extensively used in previous methods for single image dehazing [Fat08; Tan08; TH09; He+11; Men+13; Fat14; BA+16; Che+16b]. These works either separately or jointly estimate the transmission map $t(x)$ and the atmospheric light A to generate the clear image from a hazy image. Due to the under-constrained nature of single image dehazing, the success of previous methods often relies on hand-crafted priors such as dark channel prior [He+11], contrast color-lines [Fat14], color attenuation prior [Zhu+15], and non-local pior [BA+16]. However, it is difficult for these priors to be always satisfied in practice. For example, dark channel prior is known to be unreliable for areas that are similar to the atmospheric light.

More recent works learn convolutional neural networks (CNNs) to estimate components in the atmospheric scattering model for image dehazing [Cai+16; Ren+16; Li+17a; Li+18a; ZP18; Yan+18]. These methods are often trained with limited (synthetic) images, and use only a few layers of convolutional filters. The learned shallow networks have limited capacity to represent or process images, making them difficult to surpass the prior-based methods. In contrast, training deep neural networks with large-scale data has made significant progress and achieved state-of-the-art performance in many vision tasks [Kri+12; SZ14; He+16]. Moreover, the deep features extracted by a pre-trained deep network are used as pow-

erful image representation in many applications, such as domain invariant recognition [Don+14], perceptual evaluation [Zha+18a], and characterizing image statistics [Gat+16]. More recently, the architecture of CNNs itself has been recognized as a prior for image processing [Uly+17a]. In this paper, we study how to release the power of *deep* network for single image dehazing.

We propose an encoder-decoder architecture as an end-to-end system for single image dehazing. We exploit the representation power of deep features by adopting the convolutional layers of the deep VGG net [SZ14] as our encoder, and pre-train the encoder on large-scale image classification task [Rus+15]. We add skip connections with instance normalization between the encoder and decoder, and then train decoder with both ℓ_2 reconstruction loss and VGG perceptual loss [Zha+18a]. We show that the recently proposed instance normalization [Uly+16a], which is designed for image style transfer, is also effective in image dehazing. The proposed method effectively learns the statistics of clear images based on the deep feature representation, which benefits the dehazing process on the input image. Our approach outperforms the state-of-the-art results by a large margin on a recently released benchmark dataset [Li+17b], and performs surprisingly well in several cross-domain experiments. Our method depends on neither the explicit atmospheric scattering model nor the hand-crafted image priors, and only exploits the deep network architecture and pre-trained models to tackle the under-constrained single image dehazing problem. Our simple yet effective network can serve as a strong baseline for future study in this topic.

7.2.2 Related work

Traditional methods focus on representing human knowledge as priors for image processing. Tan [Tan08] assumes higher contrast of clear images and proposes a patch-based contrast-maximization method. Fattal [Fat08] assumes the transmission and surface shading are locally uncorrelated, and estimates the albedo of the scene. Dark channel prior (DCP) [He+11] assumes local patches contain low intensity pixels in at lease one color channel and hence estimates the transmission map. Fast visibility restoration (FVR) [TH09] is a filtering approach by atmospheric veil inference and corner preserving smoothing. Meng et al. [Men+13] uses boundary constraint and contextual regularization (BCCR), and Chen et al. [Che+16b] uses gradient residual minimization (GRM) to surpress artifacts. Tang et al. [Tan+14] combines priors by learning with random forests model. Color attenuation prior (CAP) [Zhu+15] assumes a linear model of brightness and the saturation and then learns the coefficients. Berman and Avidan [BA+16] assumes each color cluster in the clear image becomes a line in RGB space, and proposes non-local image dehazing (NLD).

There is an increasing interest in applying convolutional neural networks (CNNs) for image dehazing. DehazeNet [Cai+16] and multi-scale convolutional neural networks (MSCNN) [Ren+16] are trained to estimate the transmission map. AOD-Net[Li+17a] estimates a new variable based on the transformation of the atmospheric scattering model. Zhang et al. [Zha+17b] and Zhang and Patel [ZP18] and Li et al. [Li+18a] estimate transmission map and atmospheric light by sepa-

rate CNNs. Yang et al. [Yan+18] adversarially train generators for components of the atmospheric scattering model. Ren et al. [Ren+18] train network to fuse three derived inputs from an original hazy image. These methods use relatively small CNNs and do not exploit the pre-trained deep networks for image representation. A few days before our submission, we notice a preprint [Che+18] that also uses the pre-trained deep networks. The proposed method is quite different from [Che+18]: we use encoder-decoder with skip connections, while Cheng et al. [Che+18] only use feature maps extracted from one layer of the pre-trained network as input; we study instance normalization and demonstrate its effectiveness; we train an end-to-end system from hazy image to clear image, while Cheng et al. [Che+18] estimate transmission map and atmospheric light; we can generate impressive results without explicitly applying the atmospheric scattering model.

Deep neural networks can be used as "priors" for image generation and image processing. The architecture of CNNs itself can be a constraint for image processing [Uly+17a] and image generation [KW13; Goo+14a]. A pre-trained deep networks can be used as general purpose feature extractors [Don+14] and perceptual metric [Zha+18a]. The second-order information of the features extracted by a pre-trained network describes the style of images [Gat+16]. Instance normalization layers that effectively change the statistics of deep features are widely used for image style transfer [Uly+16a; Dum+17; Ghi+17; HB17; Xu+19a]. Image translation tasks with adversarial networks are often trained with batch normalization and batch size one [Iso+17], which may suffer from the statistics mismatch between training and testing.

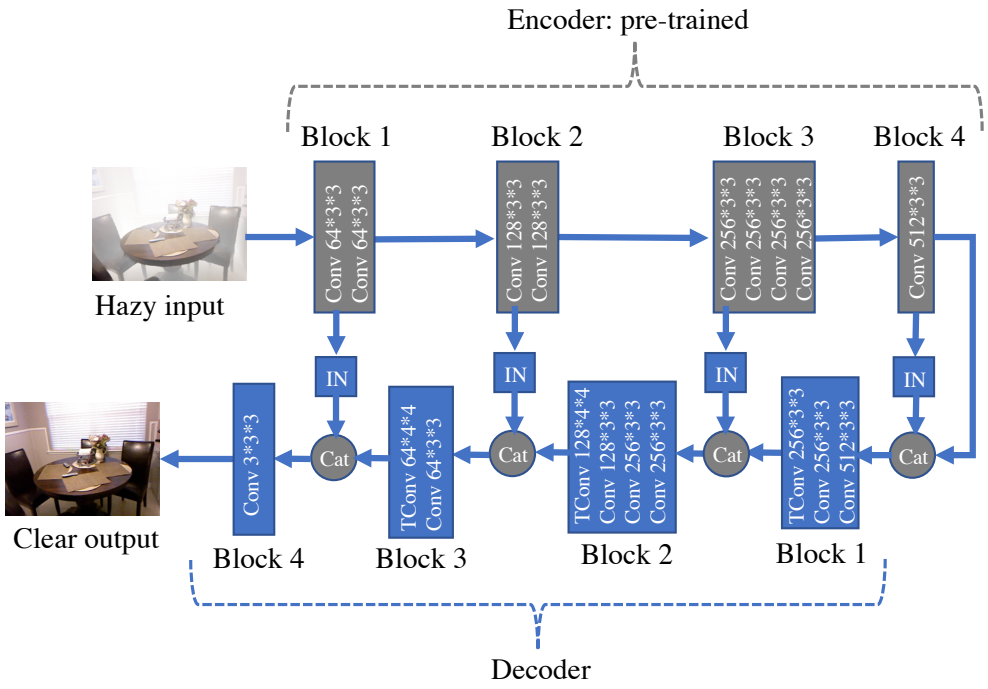


Figure 7.9: The proposed network: encoder-decoder with skip connections and instance normalization (IN); convolutional layers of pre-trained VGG [SZ14] are used as encoder; ℓ_2 reconstruction loss and VGG perceptual loss are used for training decoder and IN layers.

7.2.3 VGG-based U-Net with instance normalization

We propose an end-to-end encoder-decoder network architecture for single image dehazing, as shown in Fig. 7.9. The input is a hazy image, and the output is the desired clear image. We introduce different components of the network in the following paragraphs of this section.

Encoder. Our encoder uses the convolutional layers of the VGG net [SZ14] pre-trained on Imagenet large-scale image classification task [Rus+15]. VGG net contains five blocks of convolutional layers, and we use the first three blocks and the first convolutional layer of the forth block. Each block contains several convolutional layers, and each convolutional layer is equipped with ReLU [Kri+12] as activation function. The width (number of channels) and size (height and width) of convolutional layers are shown in Fig. 7.9. There is a maxpooling layer of stride two between blocks, which enlarges the receptive field of higher layers. The width of convolutional layer is doubled after the subsampling of feature maps by maxpooling.

The pre-trained VGG net is a powerful feature extractor for perceptual metric [Zha+18a] and image statistics [Gat+16]. Our encoder is deep and wide, and the extracted deep features are capable to capture the semantic information of the input image. We fix the encoder during training to exploit the power of pre-trained VGG net as "priors", and avoid overfitting from relatively small number of samples in image dehazing dataset.

Decoder and skip connection. Our decoder is designed to be roughly symmetric to the encoder. The decoder also contains four blocks, and each block

contains several convolutional layers. The last layer of the first three blocks of the decoder uses transposed convolutional layer to upsample the feature maps. We use ReLU activation for convolutional and transposed convolutional layers except for the last layer, where we use Tanh as activation function.

We add skip connections from the output of the first convolutional layer of encoder block 1,2,3 to the input of decoder block 4,3,2 by concatenating (cat) the feature maps, respectively. Hence our deep encoder-decoder network has a U-Net [Ron+15; Iso+17] structure except that our skip connections are based on blocks instead of layers . We use trainable instance normalization for skip connections, and have instance normalization before each convolutional layer in decoder except the first one. Our deep encoder-decoder network has large capacity, and skip connections make the information smoothly flow to easily train a large network [He+16; Li+18b].

Instance normalization. We briefly review instance normalization [Uly+16a], and discuss our motivation in applying instance normalization for single image dehazing. Let $x \in \mathbb{R}^{N\times C\times H\times W}$ represent the feature map of a convolutional layer from a minibatch of samples, where N is the batch size, C is the width of the layer (number of channels), H and W are height and width of the feature map. x_{nchw} denotes the element at height h, width w of the cth channel from the nth sample, and instance normalization layer can be written as,

$$\begin{aligned} IN(x_{nchw}) &= \gamma_{nc}\left(\frac{x_{nchw}-\mu_{nc}}{\sigma_{nc}}\right)+\beta_{nc}, \text{ where} \\ \mu_{nc} &= \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{nchw}, \ \sigma_{nc} = \sqrt{\frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W}(x_{nchw}-\mu_{nc})^2+\epsilon}, \end{aligned} \tag{7.8}$$

γ_{nc}, β_{nc} are learnable affine parameters, ϵ is a very small constant, and μ_{nc}, σ^2_{nc} represent the mean and variance for each feature map per channel per sample.

If we replace instance level variables $\gamma_{nc}, \beta_{nc}, \mu_{nc}, \sigma^2_{nc}$ with batch level variables $\gamma_c, \beta_c, \mu_c, \sigma^2_c$ that are estimated for all samples of a minibatch, we get the well-known batch normalization layer [IS15]. We show instance normalization is preferred than batch normalization for single image dehazing in our experimental ablation study.

The learnable affine parameters γ_{nc}, β_{nc} of instance normalization shift the first and second order statistics (mean and variance) of the feature maps. Instance normalization is effective for image style transfer, and the style of images can be represented by learned affine parameters [Dum+17]. Shifting the statistics of deep features extracted by pre-trained networks has achieved impressive results for arbitrary style transfer [HB17]. Shifting the statistics of images is intuitive for dehazing, however, it can be difficult to decide the exact amount to change because haze depends on the unknown depth. The deep features extracted by a pre-trained VGG net contain semantic information to effectively infer depth for haze, and hence the learned affine parameters effectively shift the statistics of images. We apply instance normalization on the deep features extracted by pre-trained VGG net for single image dehazing.

Training loss. Our network is trained with both reconstruction loss and VGG perceptual loss. Denoting the training pairs of hazy image and clear image as

$(I_n, T_n), n = 1, \ldots, N$, we use the mean squared loss,

$$\min_F \frac{1}{N} \sum_{n=1}^{N} \|F(I_n) - T_n\|^2 + \lambda \|g(F(I_n)) - g(T_n)\|^2, \tag{7.9}$$

where F represents the trainable instance normalization and decoder layers in our network, g represents the perceptual function, and λ is a hyperparameter. We set $\lambda = 1$, and use the features extracted by the first convultional layer of the third block from the pre-trained VGG net as perceptual function.

7.2.4 Experiments

In this section, we conduct various experiments on both synthetic and natural images to demonstrate the effectiveness of the proposed method. The atmospheric scattering model is widely used to synthesize images for both training and testing. The hazy images are synthesized from groundtruth clear images and grountruth depth images [Li+17b; Anc+16], or estimated depth images [Sak+17].

We train our model on the recently released RESIDE-standard dataset [Li+17b]. RESIDE-standard contains 13,990 images for training, and 500 images for testing. These images are generated by existing indoor depth datasets, NYU2 [Sil+12] and Middlebury stereo [Sch+14]. The atmospheric scattering model is used, where atmospheric lights A is randomly chosen between (0.7, 1.0) for each channel, and scattering coefficient β is randomly selected between (0.6, 1.8).

We also apply our model trained on RESIDE-standard for cross-domain evaluation on D-Hazy [Anc+16], I-Haze [Anc+18a] and O-Haze [Anc+18b] dataset.

D-Hazy dataset [Anc+16] is another synthetic dataset, which contains 23 images synthesized from Middlebury and 1449 images synthesized from NYU2, with atmospheric lights $A = (1, 1, 1)$ and scattering coefficient $\beta = 1$. Though D-Hazy dataset use the same clean images as RESIDE-standard, the generated hazy images are quite different. I-Haze [Anc+18a] and O-Haze [Anc+18b] are two recent released datasets on natural indoor and outdoor images, respectively. I-Haze contains 35 pairs of indoor images and O-Haze contains 45 pairs of outdoor images, where the hazy images are generated by using a physical haze machine.

We compare our results quantitatively and qualitatively with previous methods. We compare with prior-based methods, DCP [He+11], FVR [TH09], BCCR [Men+13] , GRM [Che+16b], CAP [Zhu+15] and NLD [BA+16] . We also compare with learning-based methods DehazeNet [Cai+16], MSCNN [Ren+16] , and AOD-Net [Li+17a]. We have provided a brief review of these baseline methods in Section 7.2.2. We use peak signal-to-noise ratio (PSNR) and structural similarity (SSIM) as metrics for quantitative evaluation. For the benchmark evaluation on RESIDE-side, all the learning-based methods are trained on the same dataset. For cross-domain evaluation on D-Hazy, O-Haze and I-Haze, we use the released best pre-trained model for the learning-based baseline methods.

We train our model by SGD with minibatch size 16 and learning rate 0.1 for 60 epochs, and linearly decrease the learning rate after 30 epochs. We use momentum 0.9 and weight decay 10^{-4} for all our experiments. We will release our Pytorch code and pre-trained models [1].

[1] `https://github.com/nightldj/dehaze_release`

	DCP	FVR	BCCR	GRM	CAP
PSNR	16.62	15.72	16.88	18.86	19.05
SSIM	0.8179	0.7483	0.7913	0.8553	0.8364
	NLD	DehazeNet	MSCNN	AOD-Net	**Ours**
PSNR	17.29	21.14	17.57	19.06	**27.79**
SSIM	0.7489	0.8472	0.8102	0.8504	**0.9556**

Table 7.3: Quantitative results on RESIDE-standard dataset [Li+17b].

7.2.4.1 Quantitative evaluation on benchmark dataset

We present the performance of our network and baseline methods on the RESIDE-standard benchmark dataset [Li+17b] in Table 7.3. Our network and the learning-based baselines [Cai+16; Ren+16; Li+17a] are trained on the provided synthetic data, and evaluated on the separate testing set. We evaluate our results by metrics provided by [Li+17b], and compare with the baseline results reported in [Li+17b]. The learning-based methods perform slightly better than the prior-based method. CAP [Zhu+15] performs best in prior-based method, which has a learning phase for the coefficients of the linear model. DehazeNet [Zhu+15] performs best in baseline methods, which uses a relatively small network to predict components.

Our approach outperforms all the baseline methods on both PSNR and SSIM by a large margin. The synthetic data for both training and testing are generated by the atmospheric scattering model, and the baseline methods explicitly use the atmospheric scattering model. In contrast, our approach only uses instance normalization to transform the statistics of deep features . The superior performance of our network on the benchmark dataset demonstrate the effectiveness of *deep* networks and instance normalization for single image dehazing.

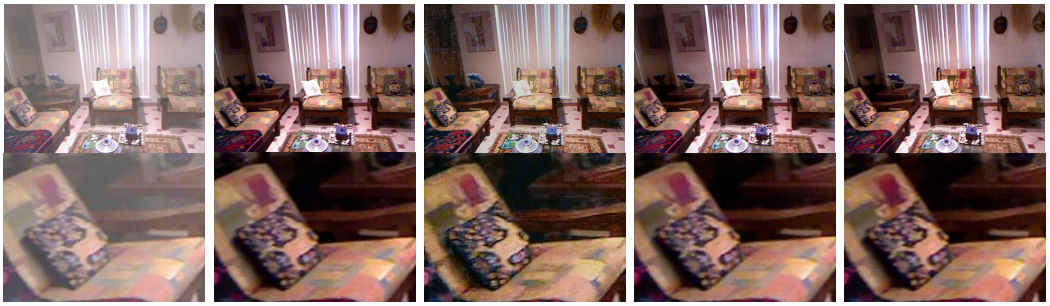


Figure 7.10: An example of qualitative results in ablation study. We zoom in the bottom left corner of the images to show more details in the second row.

Skip	NA	BN	IN	NA	BN
Dec	NA	NA	NA	BN	BN
PSNR	18.24	25.67	26.00	25.99	26.38
SSIM	0.7945	0.9442	0.9414	0.9385	0.9519

Skip	IN	NA	BN	IN	Perceptual
Dec	BN	IN	IN	IN	loss
PSNR	26.89	26.57	27.67	<u>27.75</u>	**27.79**
SSIM	0.9535	0.9381	0.9543	<u>0.9549</u>	**0.9556**

Table 7.4: Ablation study on RESIDE-standard dataset.

7.2.4.2 Ablation study

We provide more discussion on the proposed network. We verify the effectiveness of instance normalization with ablation study on network structures, as shown in Table 7.4. We use no normalization (NA), batch normalization (BN), or instance normalization (IN) for skip connections and decoders, respectively. The normalization layers are added before each convolutional layer of the decoder except for the first layer. All the results in Table 7.4 are obtained by only using reconstruction loss ($\lambda = 0$ in loss function (7.9)) except for the last one, where IN and combined loss ($\lambda = 1$) are used. We train and evaluate our network on the RESIDE-standard dataset.

First, comparing the NA results in Table 7.4 with previous best results in Table 7.3, our encoder-decoder only achieves competitive results. Second, adding normalization to either skip connections or decoder significantly improves the performance of our network. The normalization layers for decoder are implicitly applied to the features from the skip connections, which makes the result of only normalizing decoder slightly better than only normalizing skip connections. Third, instance normalization works better than batch normalization, which demonstrates the effectiveness of shifting the mean and variance of deep features at instance level.

Finally, the perceptual loss only helps a little for quantitative evaluation, but it can help generate more visually appealing output images. We show an qualitative example in Fig. 7.10, where the hazy input, the groundtruth clear image, outputs of our network without normalization layers and no perceptual loss (NA-NA), our network with instance normalization and no perceptual loss (IN-IN), and our network with instance normalization and perceptual loss (IN-IN-Percep). We enlarge the bottom left corner of the results to show more details. The results of IN-IN look much better than NA-NA. The enlarged area of the result with perceptual loss (IN-IN-Percep) looks sharper and clearer.

7.2.4.3 Cross-domain evaluation

In this section, we focus on the cross-domain performance by evaluating our network trained on RESIDE-standard [Li+17b] on the cross domain datasets, D-Hazy [Anc+16], I-Haze [Anc+18a] and O-Haze [Anc+18b]. We com-

	D-Hazy-NYU		D-Hazy-MB		I-Haze		O-Haze	
	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM	PSNR	SSIM
DCP	11.56	0.6695	12.13	0.6752	13.41	0.4930	17.01	0.4875
CAP	13.29	0.7266	14.36	**0.7526**	15.27	0.5603	16.68	0.4810
DehazeNet	13.02	0.7256	13.78	0.7342	**16.73**	0.6263	**17.90**	**0.5514**
MSCNN	13.67	0.7413	13.97	0.7488	15.93	0.5896	16.27	0.4947
AOD-Net	12.44	0.7147	13.48	0.7470	15.00	0.5828	16.22	0.4142
Ours	**18.11**	**0.8268**	**15.63**	0.7338	16.04	**0.6332**	17.46	0.5337

Table 7.5: Quantitative results for cross-domain evaluation.

pare with baseline methods that have publicly available code, and these are strong baselines according to benchmark evaluation in Table 7.3. For learning-based methods DehazeNet [Cai+16], MSCNN [Ren+16], and AOD-Net [Li+17a], we use the best model the authors have released. MSCNN [Ren+16] and AOD-Net [Li+17a] are trained with synthetic images similar to RESIDE-standard, while DehazeNet [Cai+16] is trained with patches of web images.

We present the quantitative results in Table 7.5, where we use bold to label the best results and underline to label the second best results. Our approach achieves best results, or close to the best results for all the cross-domain evaluations. Our first observation is that the learning-based methods [Cai+16; Ren+16; Li+17a], including ours, generalize reasonably well and perform equally or better than the prior-based methods [He+11; Zhu+15].

Our network performs well on the cross-domain D-Hazy dataset [Anc+16]. Particularly, our approach outperforms all baseline methods by a large margin on the images synthesized from NYU depth dataset. D-Hazy dataset is synthesized by the same clear images as our training data RESIDE-standard, but uses different parameters of the atmospheric scattering model. Our trained network has effectively

captured the statistics of the deep features of the desired clear images.

I-Haze [Anc+18a] and O-Haze [Anc+18b] images look quite different from our training images, and our network may have difficulty to infer the exact statistics of deep features for these images. DehazeNet [Cai+16] may have gained some advantage on these two datasets because it is trained on patches of web images. Our approach still produces competitive results compared with DehazeNet [Cai+16], and outperforms all the other baselines. Notice again that our network does not use the powerful atmospheric scattering model, and is only trained on a limited number of indoor synthetic images. The cross-domain evaluation further demonstrates the power of *deep* features and instance normalization in our approach.

7.2.4.4 Qualitative evaluation

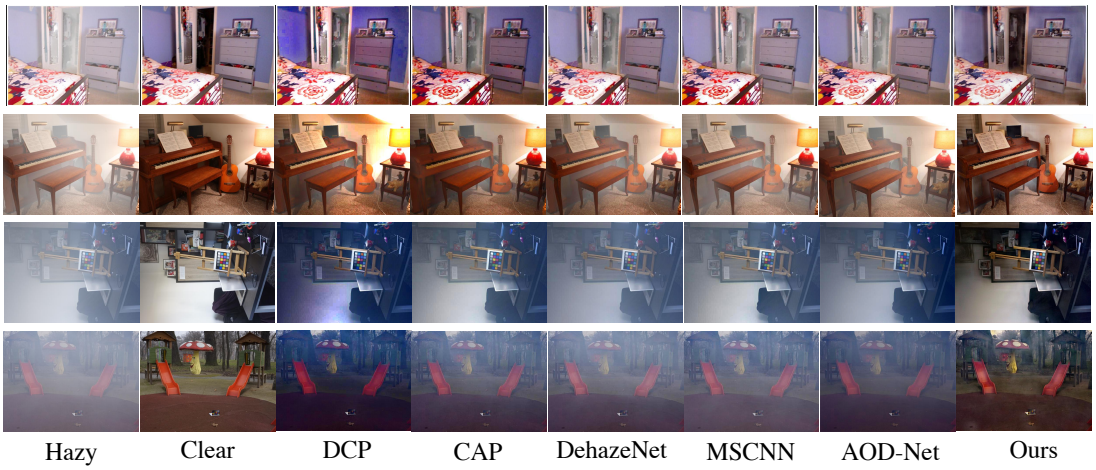


Figure 7.11: Qualitative evaluation on cross-domain dataset. The four examples are from D-Hazy-NYU [Anc+16], D-Hazy-MB [Anc+16], I-Haze [Anc+18a] and O-Haze [Anc+18b], respectively. Best viewed in color and zoomed in.

We present qualitative results from cross-domain evaluation in Fig. 7.11. The images are from D-Hazy-NYU [Anc+16], D-Hazy-MB [Anc+16], I-Haze [Anc+18a]

and O-Haze [Anc+18b], respectively. We show the hazy image and groundtruth clear image, and compare our results with DCP [He+11], CAP [Zhu+15], DehazeNet [Cai+16], MSCNN [Ren+16], and AOD-Net [Li+17a]. We use the best released model for the learning-based baselines [Cai+16; Ren+16; Li+17a], and train our network on RESIDE-standard [Li+17b].

Our network makes the best efforts to remove haze and recover the real color of images, as shown in Fig. 7.11. The results of baselines still have a large amount of undesired haze and look blurry (row 2,3,4). Particularly, the baselines have difficulty in dark areas of the image, and DCP also has difficulty in area of white and blue walls (row 1,3). For the outdoor image (row 4), our network produces a little artifact due to the significant domain difference between the desired images and the training indoor images. Use regularizers such as total variation [Rud+92] may help reduce these artifacts, and we plan to investigate it in the future. Our simple yet effective network has generated visually appealing results, without depending on extra constraints like the atmospheric scattering model.

7.2.5 Discussion

We proposed a simple yet effective end-to-end system for single image dehazing. Our network has an encoder-decoder architecutre with skip connections. We manipulated the statistics of deep features extracted by pre-trained VGG net and demonstrated the effectiveness of instance normalization for image dehazing. Moreover, without explicitly using the atmospheric scattering model, our approach

outperforms previous methods by a large margin on the benchmark datasets. Notice that both the training and testing data are generated by the atmospheric scattering model, and the baseline methods all explicitly use the model. Our network effectively learns the transformation from hazy image to clear image with limited synthetic data, and generalizes reasonably well.

The atmospheric scattering model is powerful and has been successfully deployed for image dehazing in the past decade. However, the atmospheric scattering model, as a simplified model, also constrained the learnable components to be "linearly" combined by element-wise multiplication and summation, which may not be ideal for training deep models. Our study sheds light on the power of *deep* neural networks and the *deep* features extracted by pre-trained network for single image dehazing, and encourages the rethinking on how to effectively exploit the physical model for haze. How will physical model help when training powerful deep networks? It is still an open question, and our approach serves as a strong baseline for future study.

Our network outperforms state-of-the-art methods by a large margin on the benchmark dataset, and achieves competitive results on cross-domain evaluation. The key idea of our approach is to apply instance normalization to shift the statistics of deep features for image dehazing. For cross-domain evaluation, it may be difficult to effectively infer the desired statistics of deep features of clear images that is quite different from the training data. Our generalization ability can be significantly improved by training from large-scale natural images. In the future, we will explore adversarial training to use unpaired hazy and clear images that are easier to collect

from the web.

7.2.6 GAN-based Loss without Paired Training Images

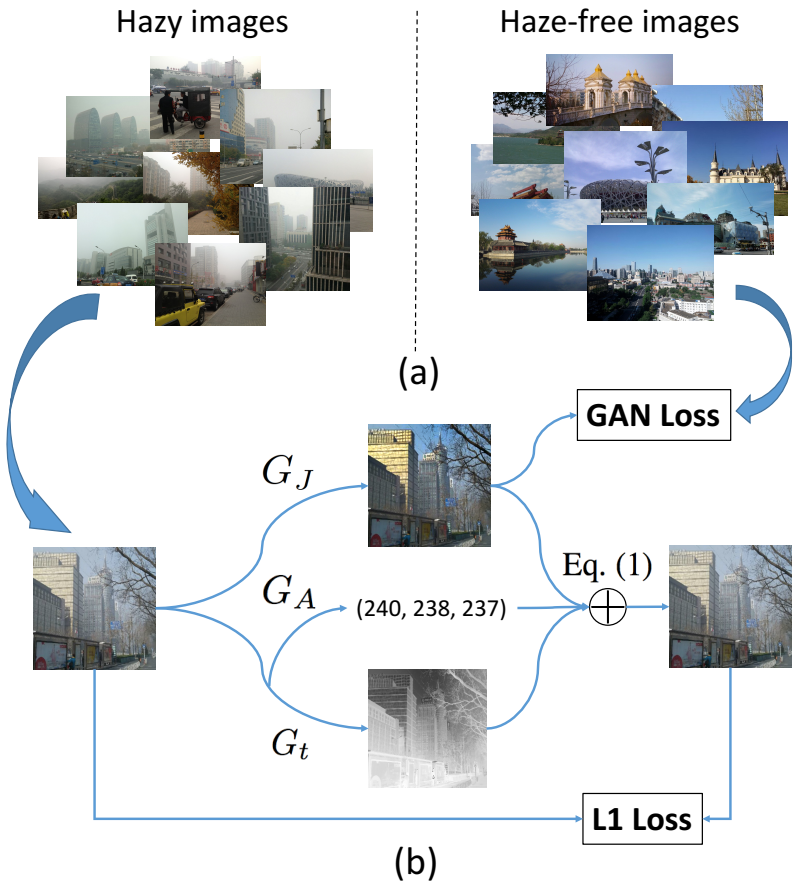


Figure 7.12: (a) Unpaired dataset with natural hazy images and haze-free images. (b) Overall architecture of our Disentangled Dehazing Network. G_J, G_t, G_A indicate the generators for the scene radiance, the medium transmission and the global atmosphere light, respectively.

Single image dehazing is a challenging under-constrained problem because of the ambiguities of unknown scene radiance and transmission. Many methods solve this problem using various hand-designed priors or by supervised training on synthetic hazy image pairs. In practice, however, the pre-defined priors are easily violated and the paired image data is unavailable for supervised training. We further

propose *Disentangled Dehazing Network*, an end-to-end model that generates realistic haze-free images using only unpaired supervision. Our approach alleviates the paired training constraint by introducing a physical-model based disentanglement and reconstruction mechanism. A multi-scale adversarial training is employed to generate perceptually haze-free images. Experimental results on synthetic datasets demonstrate our superior performance compared with the state-of-the-art methods in terms of PSNR, SSIM and CIEDE2000. Through training on purely natural haze-free and hazy images from our collected HazyCity dataset, our model can generate more perceptually appealing dehazing results.

We present our GAN-based architecture in Fig. 7.12 for training with unpaired natural and hazy images. We refer readers to [Yan+18] for more details on our GAN-based dehazing.

Chapter 8: Knowledge Distillation with Conditional Adversarial Networks

We have applied GAN framework for image processing in Chapter 7, which is one of the most popular tasks since GAN was proposed. In this chapter, we show GAN framework can be used for tasks besides image processing and generation. We use conditional adversarial network to design effective loss for knowledge distillation to transfer knowledge from a pre-trained large teacher network to train a small student network. The small student network is fast during inference, and can be easier to deploy to devices with limited computing power. We study the trade-off between accuracy and acceleration, and the proposed network can achieve 7× acceleration without loss of accuracy. We now introduce our GAN-based knowledge distillation presented in [Xu+18b].

8.1 Introduction

Deep neural networks (DNNs) achieve massive success in artificial intelligence by substantially improving the state-of-the-art performance in various applications. The accuracy of DNNs for large-scale image classification has become comparable to humans on several benchmark datasets [Rus+15]. The recent progress towards

such impressive accomplishment is largely driven by exploring deeper and wider network architectures [He+16; ZK16]. However, it is difficult to deploy the trained modern networks on embedded systems for real-time applications because of the heavy computation and memory cost. In the meantime, the demand for low cost networks is increasing for applications on mobile devices and autonomous cars.

Do DNNs really need to be deep and wide? Early theoretical studies suggest that shallow networks are powerful and can approximate arbitrary functions [Cyb89; Hor+89]. More recent theoretical results show depth is indeed beneficial for the expressive capacity of networks [ES16; Tel16; LS17; SS17]. Moreover, the overparameterized and redundant networks, which can easily memorize and overfit the training data, surprisingly generalize well in practice [Zha+17a]. Various explanations have been investigated, but the secret of deep and wide networks remains an open problem.

Empirical studies suggest that the performance of shallow networks can be improved by learning from large networks following the student-teacher strategy [BC14; Urb+17]. In these approaches, the student networks are forced to mimic the output probability distribution of the teacher networks to transfer the knowledge embedded in the soft targets. The intuition is that the *dark knowledge* [Hin+15], which contains the relative probabilities of "incorrect" answers , is informative and representative. For example, we want to classify an image over the label set (dog, cat, car). Given an image of a dog, a good teacher network may mistakenly recognize it as cat with small probability, but should seldom recognize it as car; the soft target of output distribution over categories for this image, $(0.7, 0.3, 0)$, contains

more information such as categorical correlation than the hard target of one-hot vector, $(1, 0, 0)$. The student is trained by minimizing a predetermined loss which measures similarity between student and teacher output, such as Kullback-Leibler (KL) divergence.

In previous studies, knowledge transfer has been used to train shallow but wide student networks, which potentially have more parameters than the teacher networks [BC14; Urb+17]; ensemble of networks are used as teacher, and a student network with similar architecture and capacity can be trained [Hin+15]; particularly, a small deep and thin network is trained to replace a shallow and wide network for acceleration [Rom+15], given the best teacher at that time is the shallow and wide VGGNet [SZ14]. Since then, the design of network architecture has advanced. ResNet [He+16] has significantly deepened the networks by introducing residual connections, and wide residual networks (WRNs) [ZK16] suggest widening the networks leads to better performance. It is unclear whether the dark knowledge from the state-of-the-art networks based on residual connections, which are both deep and wide, can help train a shallow and/or thin network (also with residual connections) for acceleration.

In this paper, we focus on improving the performance of a shallow and thin modern network (student) by learning from the dark knowledge of a deep and wide network (teacher). Both the student and teacher networks are convolutional neural networks (CNNs) with residual connections, and the student network is shallow and thin so that it can run much faster than the teacher network during inference. Instead of adopting the classic student-teacher strategy of forcing the output of a

student network to exactly mimic the soft targets produced by a teacher network, we introduce conditional adversarial networks to transfer knowledge from teacher to student. We empirically show that the loss learned by the adversarial training has the advantage over the predetermined loss in the student-teacher strategy, especially when the student network has relatively small capacity.

Our learning loss approach is inspired by the recent success of conditional adversarial networks for various image-to-image translation applications [Iso+17]. We show that adversarial nets can benefit a task that is very different from image generation. In the student-teacher strategy, forcing a student network to exactly mimic one of the soft targets (or the average/ensemble of several teacher networks) is not only unnecessary (because of the multi-modality [1]), but also difficult (because the student has smaller capacity). Our approach preserves the multi-modality by introducing an auxiliary network for learning the loss to transfer the knowledge.

8.1.1 Related work

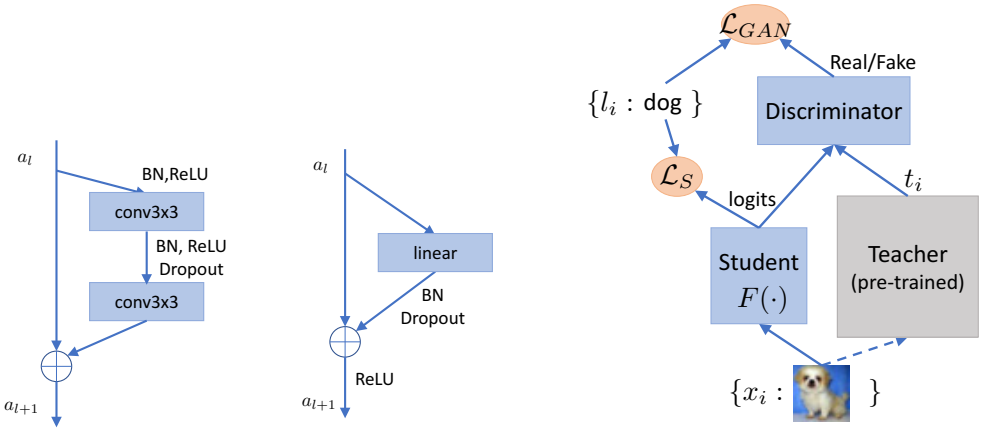
Network acceleration techniques can be roughly divided into three categories: low precision, sparse parameter pruning, and knowledge distillation. Low precision methods use limited number of bits to store and operate the network weights [Ras+16; Li+17c], which often achieve conceptual acceleration because mainstream GPUs have limited support for low precision computation. Networks can be directly modified by pruning and factorizing the redundant weights

[1] For the previous example, the output distribution for a dog image can also be (0.8, 0.2, 0). In fact, there are infinite number of soft targets that can correctly predict the label.

[How+17], which aim to construct networks of similar architecture with reduced number of weights by assuming sparsity. Moreover, network pruning papers mostly report indirect speedup measured in the number of basic operations, rather than by inference time.

Knowledge distillation is a principled approach to train small neural networks for acceleration. We slightly generalize the term *knowledge distillation* to represent all methods that train student networks by transferring knowledge from teacher networks. Bucilu et al. [Buc+06] pioneered this approach for model compression. Ba and Caruana [BC14] and Urban et al. [Urb+17] trained shallow but wide student by learning from a deep teacher, which were not primarily designed for acceleration. Hinton et al. [Hin+15] generalized the previous methods by introducing a new metric between the output distribution of teacher and student, as well as a tuning parameter. Variants of knowledge distillation has also been applied to tasks in other domains [She+16; Luo+16; Che+17b; Teh+17] A recent preprint [KK17] presented promising preliminary results on CIFAR-10 by learning a small ResNet from a large ResNet. Another line of research focuses on transferring intermediate features instead of soft targets from teacher to student [Rom+15; Wan+16; ZK17; Yim+17; HW17; You+17]. Our approach is complementary to those methods by using adversarial networks to learn a new metric between the output distribution of teacher and student.

Generative adversarial networks (GAN) has been extensively studied over recent years since [Goo+14a]. GAN trains two neural networks, the generator and the discriminator, in an adversarial learning process that alternatively up-



(a) Residual blocks for convolutional neural networks [ZK16] (left) and multi-layer perceptron (right). Blocks are equipped with batch normalization (BN), activation ReLU, and dropout. a_l is the output of the lth block.

(b) Proposed adversarial training. The deep and wide teacher is pre-trained offline. The student network and discriminator are updated alternatively. Additional supervised loss is added for both student and discriminator.

Figure 8.1: Network architectures.

dates the two networks. We use adversarial networks conditioned on input images [Iso+17; Ode+17; Xu+19a]. Unlike previous works that focused on image generation, we aim at learning a loss function for knowledge distillation, which requires quite different architectural choices for our generator and discriminator. A recent preprint [Bel+18] appears a few months later than ours has a similar approach for network compression. We are the first to apply adversarial training for knowledge distillation. Moreover, we provide systematical study on choosing the student.

8.2 Learning loss for knowledge distillation

In this section, we introduce the learning loss approach based on conditional adversarial networks. We start from a recap of modern network architectures (section 8.2.1), and then describe the dark knowledge that can be transferred from teacher to student networks (section 8.2.2). Our approach with adversarial net-

works for learning loss is detailed in section 8.2.3.

8.2.1 Neural networks with residual connection

Residual blocks are shown to be effective for training deep CNNs to achieve state-of-the-art performance [He+16; ZK16; Li+18b]. We build both student and teacher networks by stacking the residual convolutional blocks shown in Figure 8.1a (left). The first layer contains 16 filters of 3×3 convolution, followed by a stack of $6n$ layers, which is 3 groups of n residual blocks, and each block contains two convolution layers equipped with batch normalization [IS15], ReLU [Kri+12] and dropout [Sri+14]. The output feature map is subsampled twice, and the number of filters are doubled when subsampling . After the last residual block is the global average pooling, and then fully-connected layer and softmax. In the following sections, the architecture of wide residual networks (WRNs) is denoted as WRN-d-m following [ZK16], where the total depth is $d = 6n+4$, and m is the widen factor that increases the number of filters by m times in each residual block. Our teacher network is deep and wide WRN with large d and m, while student network is shallow and thin WRN with small d and m.

8.2.2 Knowledge distillation

The output of neural networks for image classification is a probability distribution over categories, which is generated by applying a softmax function over the output of the last fully connected layer (known as *logits*). Rich information

is embedded in the output of a teacher network, and we can use logits to transfer the knowledge to student network [Buc+06; BC14; Urb+17; Hin+15]. We review [Hin+15] that generalized previous methods, which provides a metric between student and teacher logits for *knowledge distillation (KD)*.

The logits vector generated by pre-trained teacher network for an input image $x_i, i = 1, \ldots, N$ is represented by t_i, where the dimension of vector $t_i = (t_i^1, \ldots, t_i^C)$ is the number of categories C. We now consider training a student network F to generate student logits $F(x_i)$. By introducing a parameter called temperature T, the generalized softmax layer can convert logits vector t_i to probability distribution q_i,

$$M_T(t_i) = q_i, \text{ where } q_i^j = \exp(t_i^j/T)/\sum_k \exp(t_i^k/T). \tag{8.1}$$

where higher temperature T produces softer probability over categories. The regular softmax for classification is a special case of the generalized softmax with $T = 1$.

Hinton et al. [Hin+15] proposed to minimize the KL divergence between teacher and student,

$$\mathcal{L}_{KD}(F, T) = 1/N \sum_{i=1}^{N} \mathrm{KL}(M_T(t_i) \| M_T(F(x_i))), \tag{8.2}$$

and show that when T is very large, $\mathcal{L}_{KD}$ becomes the Euclidean distance between teacher and student logits. Given the image-label pairs $\{x_i, l_i\}$, the cross-entropy loss for supervised training of a neural network is

$$\mathcal{L}_S(F) = 1/N \sum_{i=1}^{N} \mathcal{H}(l_i,\, M_1(F(x_i))), \tag{8.3}$$

which is widely used for standard supervised learning. Finally, Hinton et al. [Hin+15] proposed to minimize the weighted sum of $\mathcal{L}_{KD}$ and $\mathcal{L}_S$ to train a student network,

$$\mathcal{L}_1(F,T) = 1/2\mathcal{L}_S(F) + T^2\mathcal{L}_{KD}(F,T). \tag{8.4}$$

8.2.3 Learning loss with adversarial networks

Overview. The main idea of learning the loss for transferring knowledge from teacher to student is depicted in Figure 8.1b. Instead of forcing the student to exactly mimic the teacher by minimizing KL-divergence in $\mathcal{L}_1(F,T)$ of Equation (8.4), the knowledge is transferred from teacher to student through a discriminator in our approach. This discriminator is trained to distinguish whether the output logits is from teacher or student network, while the student is adversarially trained to fool the discriminator, i.e., output logits that are indistinguishable to the teacher logits.

There are several benefits of the proposed method. First, the learned loss is often effective, as has already been demonstrated for several image to image translation tasks [Iso+17]. Moreover, our approach relieves the pain for hand-engineering the loss. Though the parameter tuning and hand-engineering of the loss is replaced by hand-engineering the discriminator networks in some sense, our empirical study shows that the performance is less sensitive to the discriminator architecture than the temperature parameter in knowledge distillation. The second benefit is closely related to the multi-modality of network output. As discussed before, it is unnecessary and difficult to exactly mimic the output of teacher networks. The trained discriminator can capture the relative similarities between the categories from the multi-modal logits of teacher, and directs the student to produce correct but not

necessarily same outputs as the teacher.

Discriminator update. We now describe the proposed method in a more rigorous way. The student and discriminator in Figure 8.1b are alternatively updated in the proposed approach. Let us first look at the update of the discriminator, which is trained to distinguish teacher and student logits. We use multi-layer perceptron (MLP) as discriminator. Its building block — residual block is shown in Figure 8.1a (right). The number of nodes in each layer is the same as the dimension of logits, i.e., the number of categories C. We denote the discriminator that predicts binary value "Real/Fake" as $D(\cdot)$. To train D, we fix the student network $F(\cdot)$ and seek to maximize the log-likelihood, which is known as binary cross-entropy loss,

$$\mathcal{L}_A(D, F) = {}^1\!/_N \sum_{i=1}^{N} \Big(\log P(\text{Real}|D(t_i)) + \log P(\text{Fake}|D(F(x_i))) \Big). \tag{8.5}$$

The plain adversarial loss $\mathcal{L}_A$ for knowledge distillation, which follows the original GAN [Goo+14a], faces two major challenges. First, the adversarial training process is difficult [Yad+18]. Even if we replace the log-likelihood with advanced techniques such as Wasserstein GAN [Arj+17] or Least Squares GAN [Mao+16], the training is still slow and unstable in our experiments. Second, the discriminator captures the high-level statistics of teacher and student outputs, but the low-level alignment is missing. The student outputs $F(x_i)$ for x_i can be aligned to a completely unrelated teacher sample t_j by optimizing $\mathcal{L}_A$, which means a dog image can generate a logits vector that predicts cat. One extreme example is that the student always mispredicts dog as cat and cat as dog, but the overall output distribution may still be close to the teacher's.

To tackle these problems, we modify the discriminator objective to also predict the class labels, where the output of discriminator $D(\cdot)$ is a $C+1$ dimensional vector with C *Label* predictions and a *Real/Fake* prediction. We now maximize

$$\mathcal{L}_{\text{Discriminator}}(D, F) = 1/2(\mathcal{L}_A(D, F) + \mathcal{L}_{DS}(D, F)), \tag{8.6}$$

where $\mathcal{L}_A$ is the previously defined adversarial loss over *Real/Fake*, $\mathcal{L}_{DS}$ is the supervised log-likelihood of discriminator over *Labels*, written as

$$\mathcal{L}_{DS}(D, F) = 1/N \sum_{i=1}^{N} \Big(\log P(l_i|D(t_i)) + \log P(l_i|D(F(x_i))) \Big). \tag{8.7}$$

We assume *Label* and *Real/Fake* are conditionally independent in Equation (8.6). To avoid using this assumption, we can maximize the log-likelihood of discriminator to predict the tuple { *Label, Real/Fake* }, which requires $D(\cdot)$ to predict a $2C$ dimensional vector. In our experiments, optimizing the proposed method with or without the independent assumption achieves almost identical results. Hence we will always use the independent assumption for a more compact discriminator. Note that equation (8.6) has the same form as the auxiliary classifier GANs [Ode+17; Xu+19a].

The adversarial training becomes much more stable when the proposed discriminator also predicts category *Labels* besides *Real/Fake*. Moreover, the discriminator can provide category-level alignment between outputs of student and teacher. The student outputs of a dog image are more likely to learn from the teacher outputs that predict dogs. However, the proposed method still lacks instance-level knowledge. To further boost the performance, we start with investigating conditional

discriminators, in which the input of discriminators are logits concatenated with a conditional vector. We tried the following conditional vectors: image with convolutional embedding; label one-hot vector with embedding; and the extracted teacher logits. However, it turns out the conditional vectors are easily ignored during the training of the discriminator and does not help in practice. We will introduce a direct instance-level knowledge for training student network later.

Student update. We update the student network after updating the discriminator in each iteration. When updating the student network $F(\cdot)$, we aim to fool the discriminator by fixing discriminator $D(\cdot)$ and minimizing the adversarial loss $\mathcal{L}_A$. In the meantime, the student network is also trained to satisfy the auxiliary classifier of discriminator $\mathcal{L}_{DS}$. Besides the category-level knowledge in $\mathcal{L}_{DS}$, we introduce instance-level knowledge by aligning outputs of teacher and student,

$$\mathcal{L}_{L_1}(F) = {}^1\!/_N \sum_{i=1}^{N} \|F(x_i) - t_i\|_1. \tag{8.8}$$

The L_1 norm has been found helpful in the GAN-based image to image translation [Iso+17].

Finally, we combine the learned loss with the supervised loss $\mathcal{L}_S$ in (8.3), and minimize the following objective for the student network $F(\cdot)$,

$$\mathcal{L}_{\text{Student}}(D, F) = \mathcal{L}_S(F) + \mathcal{L}_{L_1}(F) + \mathcal{L}_{GAN}(D, F), \text{where } \mathcal{L}_{GAN}(D, F) = \frac{1}{2}(\mathcal{L}_A(D, F) - \mathcal{L}_{DS}(D, F)). \tag{8.9}$$

The sign of $\mathcal{L}_{DS}$ is flipped in (8.6) and (8.9) because both the discriminator and student are trained to preserve the category-level knowledge.

Our final loss $\mathcal{L}_{\text{Student}}(D, F)$ in (8.9) is a combination of the learned loss for

knowledge distillation and the supervised loss for neural network, and may look complicated at the first glance. However, each component of the loss is relatively simple. Moreover, since both student F and discriminator D are learned, there is no explicit parameters to be tuned in the loss function. Our experiments suggest the performance of the proposed method is reasonably insensitive to the discriminator architecture and the learned loss can outperform the hand-engineered loss for knowledge distillation.

8.3 Experiments

After presenting experimental settings, we show the benefits of our proposed method in section 8.3.1 and perform ablation study in section 8.3.2. We present the effect of depth and width of the student network in section 8.3.3, followed by the discussion of trade-off between classification accuracy and inference time in section 8.3.4.

We consider three image classification datasets: ImageNet32 [Chr+17], CIFAR-10 and CIFAR-100 [KH09], and use wide residual networks (WRNs) [ZK16] for both student and teacher networks. The teacher network is a fixed WRN-40-10, while the student network has varying depth and width in different experiments. We use multi-layer preceptron (MLP) as the discriminator in our approach. 3-layer MLP is used for most of the experiments except for section 8.3.2, in which we study the effect of discriminator depth. To speed up the experiments, the logits of teacher network are generated offline and stored in memory. We use stochastic gradient

descent (SGD) as optimizer and follow standard training scheduler, and set dropout ratio to 0.3 for both discriminator and student networks. The results below are the median of five random runs.

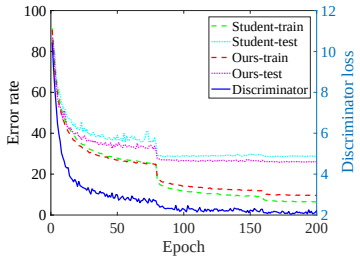
	CIFAR-10	CIFAR-100	ImageNet32
Student	7.46	28.52	48.2
Teacher	4.19	20.62	38.41
KD (T=1)	7.27	28.62	49.37
KD (T=2)	7.3	28.33	49.48
KD (T=5)	7.02	27.06	49.63
KD (T=10)	6.94	27.07	51.12
Ours	**6.09**	**25.75**	**47.39**

Table 8.1: Error rate achieved on benchmark datasets.

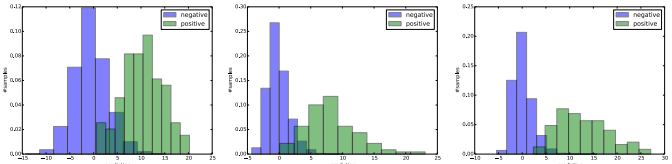
8.3.1 Benefits of learning loss

We first show the proposed method is effective for transferring knowledge from teacher to student. Table 8.1 shows the error rate of classification on the three benchmark datasets. The teacher is the deep and wide WRN-40-10. The student is much shallower and thinner, WRN-10-4 for CIFARs, and WRN-22-4 for ImageNet32. We choose a larger student network for ImageNet32 because it contains more samples and categories. We will have more discussion on wisely choosing the student architecture in sections 8.3.3 and 8.3.4 . The first two rows of Table 8.1 show the performance of standard supervised learning for student and teacher networks, without knowledge transfer. We then compare our approach with knowledge distillation (KD) in [Hin+15]. We choose the temperature parameter $T \in \{1, 2, 5, 10\}$ following the original work. No parameter is tuned for our method.

In Table 8.1, the deep and wide teacher performs much better than the shal-



(a) The training curve on CIFAR-100.



(b) The distribution of prediction for category 85 in CIFAR-100..

Figure 8.2: Analysis of the proposed method.

low and thin student with standard supervised learning, and lower bounds the error rate of the small network trained with student-teacher strategy. Baseline method KD helps the training of small networks for the two CIFARs, but does not help for ImageNet32. We conjecture the reason to be that the capacity of the student is too small to learn from knowledge distillation for larger dataset such as ImageNet32. The temperature parameter T introduced in KD is useful. For CIFARs, KD performs better when T is large, and $T = 5$ and $T = 10$ performs similarly. The proposed method improves the performance of small network for all three datasets, and outperforms KD by a margin.

8.3.2 Analysis of the proposed method

We discuss the proposed method in more details. Figure 8.2a presents the training curve of the small student network, WRN-10-4, on CIFAR-100 dataset. The loss of the discriminator (blue solid line) is gradually decreasing, which suggests the adversarial training steadily makes progress. The error rates of the proposed method for both training and testing data are decreasing. The testing error rate of the proposed method is consistently better than the pure supervised training of

Loss composition	CIFAR-10	CIFAR-100
$\mathcal{L}_S$	7.46	28.52
$\mathcal{L}_{GAN}$	14.82	47.04
$\mathcal{L}_S + \mathcal{L}_{GAN}$	6.56	27.27
$\mathcal{L}_S + \mathcal{L}_{L_1}$	6.44	26.66
$\mathcal{L}_S + \mathcal{L}_{L_1} + \mathcal{L}_{GAN}$	**6.09**	**25.75**

Table 8.2: The effect of different components of the loss in the proposed method.

Depth	1	2	3	4
Error rate	26.13	25.88	**25.75**	27.42

Table 8.3: The effect of discriminator depth on CIFAR-100.

the student model, and looks more stable between epoch 50-100. The training error rate of the proposed method is slightly worse than pure supervised learning, which suggests knowledge transfer can benefit generalization.

Next, we performing ablation study on components of the proposed approach, as shown in Table 8.2. By combining the adversarial loss and the category-level knowledge transfer (Equation (8.6)), the learned loss $\mathcal{L}_{GAN}$ performs reasonably well. However, the indirect knowledge provided by $\mathcal{L}_{GAN}$ alone is not as good as standard supervised learning $\mathcal{L}_S$. Both category-level knowledge transferred by $\mathcal{L}_{GAN}$ and instance-level knowledge transferred by $\mathcal{L}_{L_1}$ can improve the performance of training student network. Our final approach combines these components and performs the best without parameter tuning.

We present the effect of the depth of MLP as discriminator in Table 8.3. The error rate is relatively insensitive to the depth of discriminator. The error rate slightly decreases as the depth increases when the discriminator is generally shallow. When the discriminator becomes deeper, the error rate increases as the adversarial training becomes unstable. Decreasing the learning rate of discriminator sometimes helps, but it may introduce parameter tuning. The 3-layer MLP works reasonably

well and is used for all our experiments to keep the proposed method simple.

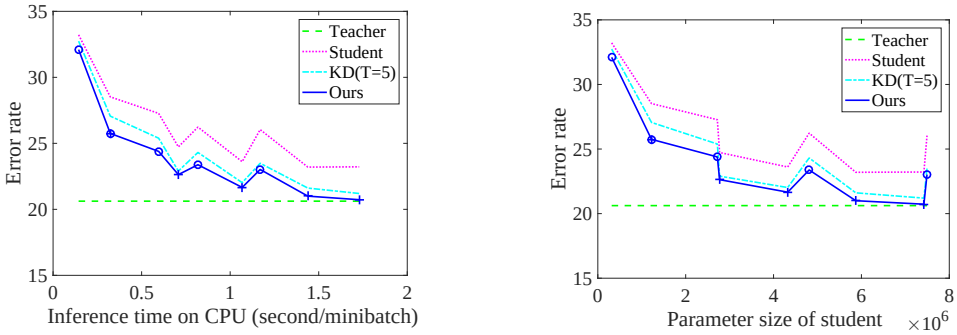
Finally, we present qualitative visualization for the proposed approach. Figure 8.2b shows the scaled histogram for the prediction of category 85 in CIFAR-100. The histogram is calculated on the 10K testing samples, in which 100 samples are from category 85 and labeled as positive (green in figure), and the other 9.9K are labeled as negative (blue in the figure). The histogram is normalized to sum up to one for positives and negatives, respectively. The three plots represent the distribution predicted by student network trained by standard supervised learning, the student network trained by the proposed approach, and the teacher network. The histogram in the middle is similar to the histogram on the right, which suggests the proposed approach effectively transfers knowledge from teacher to student.

8.3.3 Does WRN need to be deep and wide?

Urban et al. [Urb+17] asked the question for convolutional neural networks and claimed the network should at least has a few layers of convolutions. In this section, we study the modern architecture WRN of residual blocks, and show that

WRN	Size (M)	Time (s)	Student	KD (T=5)	**Ours**
10-2	0.32	0.14	33.22	32.74	**32.1**
10-4	1.22	0.32	28.52	27.16	**25.75**
10-6	2.72	0.60	27.27	25.39	**24.39**
10-8	4.81	0.82	26.23	24.31	**23.38**
10-10	7.49	1.17	26.04	23.49	**23.02**
16-4	2.77	0.71	24.73	22.9	**22.73**
22-4	4.32	1.07	23.61	22.02	**21.66**
28-4	5.87	1.44	23.2	21.61	**21.00**
34-4	7.42	1.73	23.22	21.2	**20.73**
40-10	55.9	8.73	20.62	-	**-**

Table 8.4: The effect of depth and width in student network; the parameter size, inference time and error rate on CIFAR-100.



(a) Inference time versus error rate. (b) Network size versus error rate.

Figure 8.3: Trade-off of error rate to inference time and parameter size. The figure is generated from Table 8.4. Networks WRN-10-m are labeled as circles, and WRN-d-4 are labeled as crosses for the proposed approach. The largest student is 7x smaller and 5x faster than the teacher WRN-40-10.

even for the modern architecture WRN, the network has to be deep and wide to some extent. Table 8.4 presents the results of standard supervised learning, knowledge distillation [Hin+15] and the proposed approach for different student networks trained on CIFAR-100. We first fix the depth of WRN as 10, and change the widen factor from 2 to 10. We then fix the width as 4, and increase depth from 10 to 34. The parameter size is in millions, and the inference time is measured in seconds per minibatch of 100 samples on CPU.

When the student is very small, such as WRN-10-2, it is difficult to transfer knowledge from teacher to student because the student is limited by its capacity. When the student is large, such as WRN-34-4, both KD and the proposed approach can improve the performance to approximate the teacher. The advantage of the proposed method is observed at all depths and widths but is most pronounced for relatively small students such as WRN-10-4. Increasing depth is more effective than width. For example, WRN-34-4 has less parameter than WRN-10-10, but achieves lower error rate.

8.3.4 Training student for acceleration

The shallow and thin network is much easier to deploy in practice. We present the trade-off between error rate, inference time and parameter size in Figure 8.3. The figure is generated by changing the architecture of the student network. Larger student network is more accurate but also slower. For network with similar size, such as WRN-10-10 and WRN-34-4, deeper network achieves lower error rate, while wider network runs slightly faster. When the student network is relatively large, such as WRN-34-4, the student network trained by the proposed approach can achieve competitive error rate as the teacher WRN-40-10, while being 7x smaller and 5x faster. The proposed approach also decreases the absolute error rate by 2.5% compared to the standard training without knowledge transfer.

8.4 Summarization and discussion

We study the student-teacher strategy for network acceleration in this paper. We propose to use adversarial networks to learn the loss for transferring knowledge from teacher to student. We show that the proposed approach can improve the training of student network, especially when the student network is shallow and thin. Moreover, we empirically study the effect of network capacity when adopting modern network as student and provide guidelines for wisely choosing a student to balance error rate and inference time. We can train a student that is 7x smaller and 5x faster than teacher without loss of accuracy.

The proposed approach is stable and easy to implement after applying several

advanced techniques in the GAN literature. The current implementation uses the stored logtis from teacher network to save GPU memory and computation. Generating teacher logits on the fly can be more reliable for the adversarial training. Moreover, the proposed approach can be naturally extended to use ensemble of networks as teacher. The logits of multiple teacher networks can be fed into the discriminator for better performance. We will investigate these ideas for future work.

Part III

Adversarial Training for Robustness

Chapter 9: Universal Adversarial Training

In Chapter 9 and Chapter 10, we discuss another form of minimax problem that rises from the interest of training robust machine learning models. Adversarial examples can be generated by adding small perturbation to samples, which is imperceptible to humans but will mislead the trained model to provide wrong predictions. In practice, adversarial training, which train a model based on adversarial examples generated on-the-fly, is a well-recognized method for training robust models that can defend against adversarial attacks. The adversarial training process can be formulated as optimizing a minimax problem. We discuss effective and efficient adversarial training algorithm based on [Sha+18; Sha+19; Xu+19b].

9.1 Introduction

Deep neural networks (DNNs) are vulnerable to adversarial examples, in which small and often imperceptible perturbations change the class label of an image [Sze+13; Goo+15; Ngu+15; Pap+16]. Because of the security concerns this raises, there is increasing interest in studying these attacks themselves, and also designing mechanisms to defend against them.

Adversarial examples were originally formed by selecting a single base image,

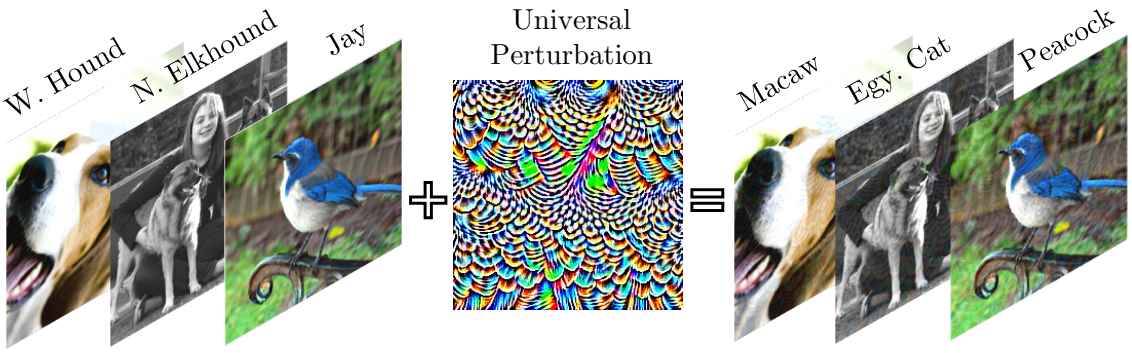


Figure 9.1: A universal perturbation made using a subset of ImageNet and the VGG-16 architecture. When added to the validation images, their labels usually change. The perturbation was generated using the proposed Algorithm 6. Perturbation pixel values lie in $[-10, 10]$ (i.e. $\epsilon = 10$).

and then sneaking that base image into a different class using a small perturbation [Goo+15; CW17b; Mad+17]. This is done most effectively using (potentially expensive) iterative optimization procedures [Don+17; Mad+17; Ath+18].

Different from per-instance perturbation attacks, Moosavi-Dezfooli et al. [MD+17b; MD+17a] show there exists "universal" perturbations that can be added to any image to change its class label (Fig. 9.1) with high probability. Universal perturbations empower attackers who cannot generate per-instance adversarial examples on the go, or who want to change the identity of an object to be selected later in the field. What's worse, universal perturbations have good cross-model transferability, which facilitates black-box attacks.

Among various methods for hardening networks to per-instance attacks, *adversarial training* [Mad+17] is known to dramatically increase robustness [Ath+18]. In this process, adversarial examples are produced for each mini-batch during training, and injected into the training data. While effective at increasing robustness, the high cost of this process precludes its use on large and complex datasets. This

cost comes from the adversarial example generation process, which frequently requires 5-30 iterations to produce an example. Unfortunately, adversarial training using cheap, non-iterative methods generally does not result in robustness against stronger iterative adversaries [Mad+17].

Contributions This paper studies effective methods for producing and deflecting universal adversarial attacks. First, we pose the creation of universal perturbations as an optimization problem that can be effectively solved by stochastic gradient methods. This method dramatically reduces the time needed to produce attacks as compared to [MD+17b]. The efficiency of this formulation empowers us to consider universal adversarial training. We formulate the adversarial training problem as a min-max optimization where the minimization is over the network parameters and the maximization is over the universal perturbation. This problem can be solved quickly using alternating stochastic gradient methods with no inner loops, making it far more efficient than per-instance adversarial training with a strong adversary (which requires a PGD inner loop). Prior to our work, it was argued that adversarial training on universal perturbations is infeasible because the inner optimization requires the generation of a universal perturbation from scratch, which requires a lot of computation and many iterations [Per+18]. We further improve the defense efficiency by providing a "free" algorithm for defending against universal perturbations. Through experiments on CIFAR-10 and ImageNet, we show that this "free" method works well in practice.

9.2 Related work

We briefly review *per-instance* perturbation attack techniques that are closely related to our paper and can be used during the universal perturbation update step of universal adversarial training. The Fast Gradient Sign Method (FGSM) [Goo+15] is one of the most popular one-step gradient-based approaches for ℓ_∞-bounded attacks. FGSM applies one step of gradient ascent in the direction of the sign of the gradient of the loss function with respect to the input image. When a model is adversarially trained, the gradient of the loss function may be very small near unmodified images. In this case, the R-FGSM method remains effective by first using a random perturbation to step off the image manifold, and then applying FGSM [Tra+17]. Projected Gradient Descent (PGD) [Kur+16b; Mad+17] iteratively applies FGSM multiple times, and is one of the strongest per-instance attacks [Mad+17; Ath+18]. The PGD version of [Mad+17] applies an initial random perturbation before multiple steps of gradient ascent. Finally, DeepFool [MD+16] is an iterative method based on a linear approximation of the training loss objective. This method formed the backbone of the original method for producing universal adversarial examples [MD+17b].

Adversarial training, in which adversarial examples are injected into the dataset during training, is an effective method to learn a robust model resistant to per-instance attacks [Mad+17; Ath+18; Hua+15; Sha+15; Sin+18]. Robust models adversarially trained with FGSM can resist FGSM attacks [Kur+16b], but can be vulnerable to PGD attacks [Mad+17]. Madry et al. [Mad+17] suggest strong

attacks are important, and they use the iterative PGD method in the inner loop for generating adversarial examples when optimizing the min-max problem. PGD adversarial training is effective but time-consuming. The cost of the inner PGD loop is high, although this can sometimes be replaced with neural models for attack generation [BF18; Pou+18; Xia+18]. These robust models are adversarially trained to fend off per-instance perturbations and have not been designed for, or tested against, universal perturbations.

Unlike per-instance perturbations, *universal* perturbations can be directly added to any test image to fool the classifier. In [MD+17b], universal perturbations for image classification are generated by iteratively optimizing the per-instance adversarial loss for training samples using DeepFool [MD+16]. In addition to classification tasks, universal perturbations are also shown to exist for semantic segmentation [Met+17]. Robust universal adversarial examples are generated as a universal targeted adversarial patch in [Bro+17]. They are targeted since they cause misclassification of the images to a given target class. Moosavi-Dezfooli et al. [MD+17a] prove the existence of small universal perturbations under certain curvature conditions of decision boundaries. Data-independent universal perturbations are also shown to exist and can be generated by maximizing spurious activations at each layer. These universal perturbations are slightly weaker than the data dependent approaches [Mop+17]. As a variant of universal perturbation, unconditional generators are trained to create perturbations from random noises for attack [RM+18a; RM+18b].

There has been very little work on defending against universal attacks. To the

best of our knowledge, the only dedicated study is by Akhtar et al., who propose a perturbation rectifying network that pre-processes input images to remove the universal perturbation [Akh+18]. The rectifying network is trained on universal perturbations that are built for the downstream classifier. While other methods of data sanitization exist [Sam+19; MC17] , it has been shown (at least for per-instance adversarial examples) that this type of defense is easily subverted by an attacker who is aware that a defense network is being used [CW17a].

Recent preprints [Per+18] model the problem of defending against universal perturbations as a two-player min-max game. However, unlike us, and similar to per-instance adversarial training, after each gradient descent iteration for updating the DNN parameters, they generate a universal adversarial example in an iterative fashion. Since the generation of universal adversarial perturbations is very time-consuming [Akh+18], this makes their approach very slow in practice and prevents them from training the neural network parameters for many iterations.

9.3 Optimization for universal perturbation

Given a set of training samples $X = \{x_i, i = 1, \ldots, N\}$ and a network $f(w, \cdot)$ with frozen parameter w that maps images onto labels, Moosavi-Dezfooli et al. [MD+17b] propose to find universal perturbations δ that satisfy,

$$\|\delta\|_p \leq \epsilon \text{ and } \text{Prob}(X, \delta) \geq 1 - \xi, \tag{9.1}$$

Algorithm 5 Iterative solver for universal perturbations [MD+17b]

Initialize $\delta \leftarrow 0$
while $\text{Prob}(X, \delta) < 1 - \xi$ **do**
  for x_i in X **do**
    if $f(w, x_i + \delta) \neq f(w, x_i)$ **then**
      Solve $\min_r \|r\|_2$ s.t. $f(w, x_i + \delta + r) \neq f(w, x_i)$
        by DeepFool [MD+16]
      Update $\delta \leftarrow \delta + r$, then project δ to ℓ_p ball
    end if
  end for
end while

$\text{Prob}(X, \delta)$ represents the "fooling ratio," which is the fraction of images x whose perturbed class label $f(w, x+\delta)$ differs from the original label $f(w, x)$. The parameter ϵ controls the ℓ_p diameter of the bounded perturbation, and ξ is a small tolerance hyperparameter. Problem (9.1) is solved by the iterative method in Algorithm 5 [MD+17b]. This iterative solver relies on an inner loop to apply DeepFool [MD+16] to each training instance, which makes the solver slow. Moreover, the outer loop of Algorithm 5 is not guaranteed to converge.

Different from [MD+17b], we consider the following optimization problem for building universal perturbations,

$$\max_{\delta} \mathcal{L}(w, \delta) = \frac{1}{N} \sum_{i=1}^{N} l(w, x_i + \delta) \text{ s.t. } \|\delta\|_p \leq \epsilon, \tag{9.2}$$

where $l(w, \cdot)$ represents the loss used for training DNNs. This simple formulation (9.2) searches for a universal perturbation that maximizes the training loss, and thus forces images into the wrong class.

The naive formulation (9.2) suffers from a potentially significant drawback;

Algorithm 6 Stochastic gradient for universal perturbation

```
for epoch = 1 ... N_ep do
  for minibatch B ⊂ X do
    Update δ with gradient variant δ ← δ + g
    Project δ to ℓ_p ball
  end for
end for
```

the cross-entropy loss is unbounded from above, and can be arbitrarily large when evaluated on a single image. In the worst-case, a perturbation that causes misclassification of just a single image can maximize (9.2) by forcing the average loss to infinity. To force the optimizer to find a perturbation that fools many instances, we propose a "clipped" version of the cross entropy loss,

$$\hat{l}(w, x_i + \delta) = \min\{l(w, x_i + \delta),\ \beta\}. \tag{9.3}$$

We cap the loss function at β to prevent any single image from dominating the objective in (9.2), and giving us a better surrogate of misclassification accuracy. In Section 9.5.2, we investigate the effect of clipping with different β.

We directly solve Eq. (9.2) by a stochastic gradient method described in Algorithm 6. Each iteration begins by using gradient ascent to update the universal perturbation δ to maximize the loss. Then, δ is projected onto the ℓ_p-norm ball to prevent it from growing too large. We experiment with various optimizers for this ascent step, including Stochastic Gradient Descent (SGD), Momentum SGD (MSGD), Projected Gradient Descent (PGD), and ADAM [KB14].

We test this method by attacking a naturally trained WideResnet 28-10 archi-

texture on the CIFAR-10 dataset. We use $\epsilon = 8$ for the ℓ_∞ constraint for CIFAR-10 following [Mad+17]. Stochastic gradient methods that use "normalized" gradients (ADAM and PGD) are less sensitive to learning rate and converge faster, as shown in Fig. 9.2. We visualize the generated universal perturbation from different optimizers in Fig. 9.3. Compared to the noisy perturbation generated by SGD, normalized gradient methods produced stronger attacks with more well-defined geometric structures and checkerboard patterns. The final evaluation accuracies (on test-examples) after adding universal perturbations with $\epsilon = 8$ were 42.56% for the SGD perturbation, 13.08% for MSGD, 13.30% for ADAM, and 13.79% for PGD. The clean test accuracy of the WideResnet is 95.2%.

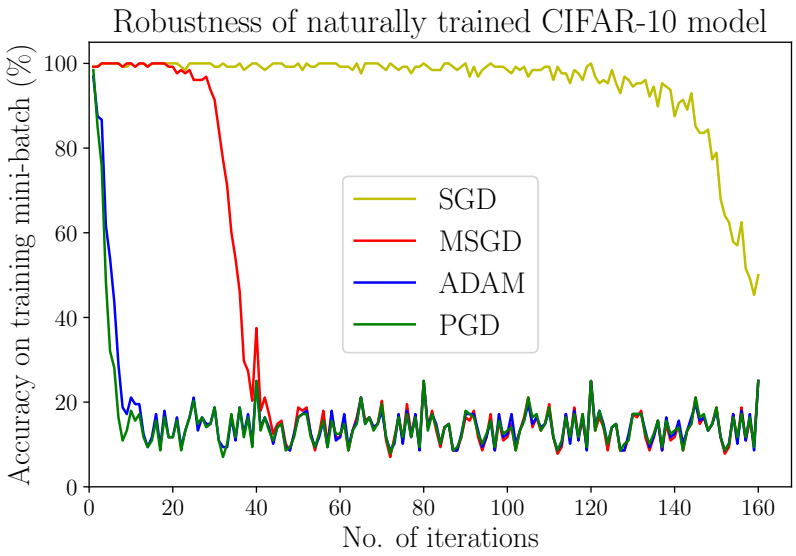


Figure 9.2: Classification accuracy on adversarial examples of universal perturbations generated by increasing the cross-entropy loss. PGD and ADAM converge faster. We use 5000 training samples from CIFAR-10 for constructing the universal adversarial perturbation for naturally trained Wide ResNet model from [Mad+17]. The batch-size is 128, ϵ=8, and the learning-rate/step-size is 1.

Our proposed method of universal attack using a clipped loss function has sev-

(a) SGD (b) MSGD (c) ADAM (d) PGD

Figure 9.3: Visualizations of universal perturbations after 160 iterations of the optimizers depicted in Fig. 9.2.

eral advantages. It is based on a standard stochastic gradient method that comes with convergence guarantees when a decreasing learning rate is used [Bot+18]. Also, each iteration is based on a minibatch of samples instead of one instance, which accelerates computation on a GPU. Finally, each iteration requires a simple gradient update instead of the complex DeepFool inner loop; we empirically verify fast convergence and good performance of the proposed method (see Section 9.5).

9.4 Universal adversarial training

We now consider training robust classifiers that are resistant to universal perturbations. Similar to [Mad+17], we borrow ideas from robust optimization. We use robust optimization to build robust models that can resist universal perturbations. In particular, we consider universal adversarial training, and formulate this problem as a min-max optimization problem,

$$\begin{aligned} \min_{w} \max_{\delta} \mathcal{L}(w, \delta) &= \frac{1}{N} \sum_{i=1}^{N} l(w, x_i + \delta) \\ \text{s.t. } \|\delta\|_p &\leq \epsilon, \end{aligned} \tag{9.4}$$

Algorithm 7 Alternating stochastic gradient method for adversarial training against universal perturbation

Input: Training samples X, perturbation bound ϵ, learning rate τ, momentum μ

```
for epoch = 1 ... N_ep do
  for minibatch B ⊂ X do
    Update w with momentum stochastic gradient
      g_w ← μ g_w − E_{x∈B}[∇_w l(w, x + δ)]
      w ← w + τ g_w
    Update δ with stochastic gradient ascent
      δ ← δ + ε sign(E_{x∈B}[∇_δ l(w, x + δ)])
    Project δ to ℓ_p ball
  end for
end for
```

where w represents the neural network weights, $X = \{x_i, i = 1, \ldots, N\}$ represents training samples, δ represents universal perturbation noise, and $l(\cdot)$ is the loss function. Here, unlike conventional adversarial training, our δ is a universal perturbation (or, more accurately, mini-batch universal). Previously, solving this optimization problem directly was deemed infeasible [Per+18], but we show that Eq. (9.4) is efficiently solvable by alternating stochastic gradient methods shown in Algorithm 7. We show that unlike [Mad+17], updating the universal perturbation only using a simple step is enough for building universally hardened networks. Each iteration alternatively updates the neural network weights w using gradient descent, and then updates the universal perturbation δ using ascent. As we show later in our experiment, the choice of the ascent optimizer does have a large impact on universal robustness.

We compare our formulation (9.4) and Algorithm 7 with PGD-based adversarial training in [Mad+17], which trains a robust model by optimizing the following

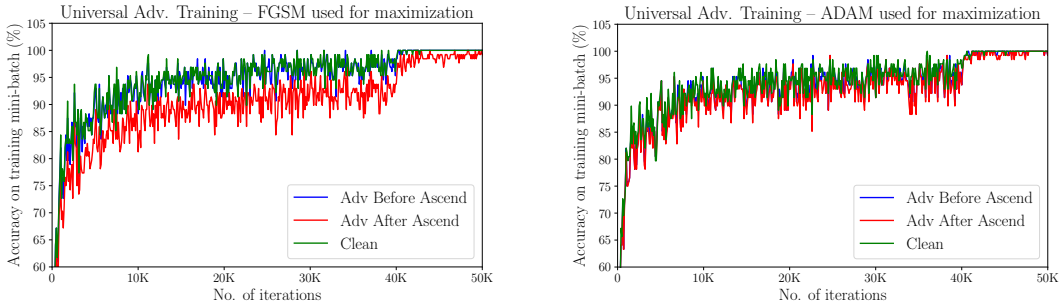


Figure 9.4: Classification accuracy for (adversarial) training of (robust) models with (top) FGSM update and (bottom) ADAM update. We show the accuracy before and after the gradient ascent for δ in Algorithm 7. We omitted the figure for SGD update because the gap between the two curves for SGD is invisible.

min-max problem,

$$\min_{w} \max_{Z} \frac{1}{N} \sum_{i=1}^{N} l(w, z_i) \text{ s.t. } \|Z - X\|_p \leq \epsilon. \tag{9.5}$$

The standard formulation (9.5) searches for per-instance perturbed images Z, while our formulation in (9.4) maximizes using a universal perturbation δ. Madry et al. [Mad+17] solve (9.5) by a stochastic method. In each iteration, an adversarial example z_i is generated for an input instance by the PGD iterative method, and the neural network parameter w is updated once [Mad+17]. Our formulation (Algorithm 7) only maintains one single perturbation that is used and refined in all iterations. For this reason, we need only update w and δ once per step (i.e., there is no expensive inner loop), and these updates accumulate for both w and δ through training.

In Fig. 9.4, we present training curves for the universal adversarial training process on the WideResnet model from [Mad+17] using the CIFAR-10 dataset. We

consider different rules for updating δ during universal adversarial training,

$$\text{FGSM } \delta \leftarrow \delta + \epsilon \cdot \text{sign}(\mathbb{E}_{x \in B}[\nabla_\delta l(w, x + \delta)]), \tag{9.6}$$

$$\text{SGD } \delta \leftarrow \delta + \tau_\delta \cdot \mathbb{E}_{x \in B}[\nabla_\delta l(w, x + \delta)], \tag{9.7}$$

and ADAM [KB14]. We found that the FGSM update rule was most effective when combined with the SGD optimizer for updating neural network weights w.

One way to assess the update rule is to plot the model accuracy before and after the ascent step (i.e., the perturbation update). It is well-known that adversarial training is more effective when stronger attacks are used. In the extreme case of a do-nothing adversary, the adversarial training method degenerates to natural training. In Fig. 9.5, we see a gap between the accuracy curves plotted before and after gradient ascent. We find that the FGSM update rule leads to a larger gap, indicating a stronger adversary. Correspondingly, we find that the FGSM update rule yields networks that are more robust to attacks as compared to SGD update (see Fig. 9.5).

Interestingly, while our universal adversarial training alg. 7 is for training models that are robust to universal perturbations, we see that when using a strong update rule, the hardened models become robust against ℓ_∞ per-instance white-box attacks generated using a 20-step PGD attack. While training with the "normalized" (FGSM and ADAM) universal perturbation update rules result in models that resist universal perturbations, the FGSM update rule produces models that are more resistant against per-instance attacks compared to the ADAM update rule. The ac-

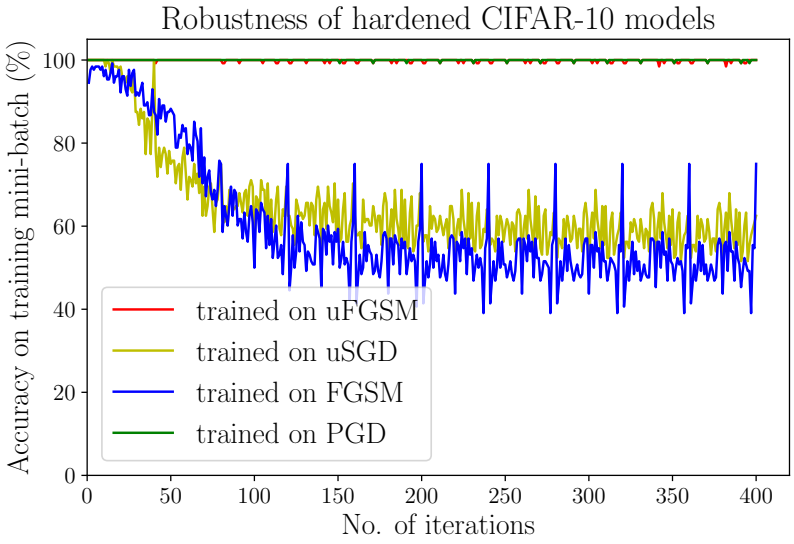
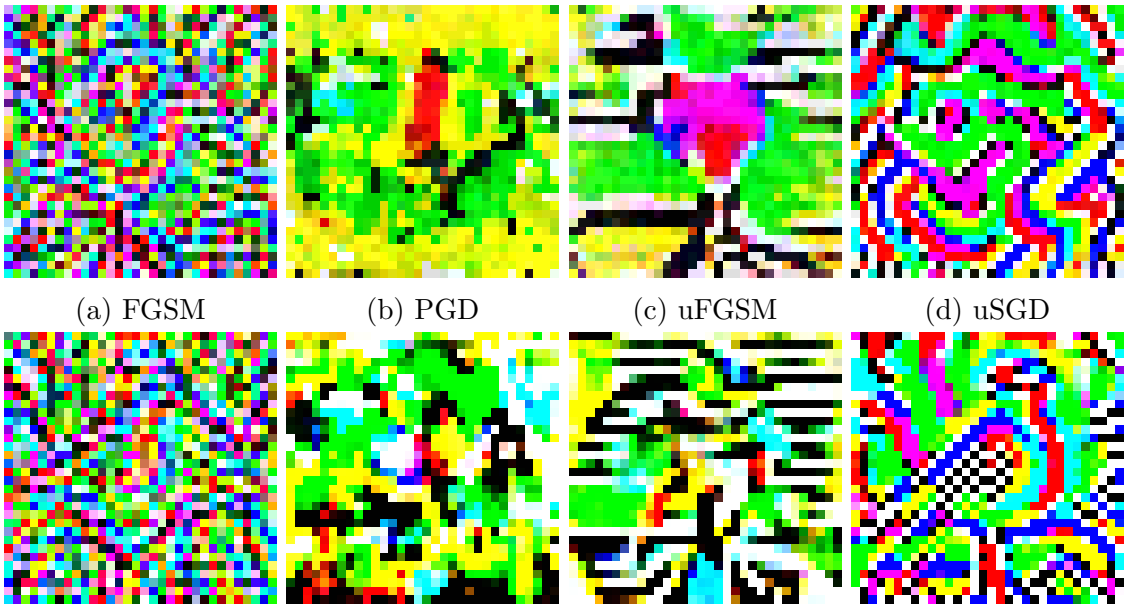


Figure 9.5: Classification accuracy on training data when the universal perturbations are updated with the ADAM optimizer. We use 5000 training samples from CIFAR-10 for constructing the universal adversarial perturbation for an adversarially trained WideResnet model from [Mad+17]. The batch-size is 128, ϵ=8, and the learning-rate/step-size is 1.

curacy of a universally hardened network against a white-box per-instance PGD attack is 17.21% for FGSM universal training, and only 2.57% for ADAM universal training. When compared to FGSM per-instance adversarial training, which has comparable computation cost, the universally robust model is even more robust against per-instance attacks! FGSM per-instance adversarial training achieves 0.00% accuracy on per-instance adversarial examples built using the same PGD attack setting. More per-instance comparisons are provided in the supplementary material.

9.4.1 Attacking hardened models

We evaluate the robustness of different models by applying Algorithm 6 to try to find universal perturbations. We attack universally adversarial trained mod-



(a) FGSM (b) PGD (c) uFGSM (d) uSGD

(e) FGSM (f) PGD (g) uFGSM (h) uSGD

Figure 9.6: The universal perturbations made using PGD and ADAM for 4 different robust models trained on CIFAR-10: adversarially trained with FGSM or PGD, and universally adversarially trained with FGSM (uFGSM) or SGD (uSGD). Perturbations were made using 400 iterations. The top row perturbations are made using PGD and the bottom row perturbations are made using ADAM.

els (produced by Eq. (9.4)) using the FGSM universal update rule (uFGSM), or the SGD universal update rule (uSGD). We also consider robust models from per-instance adversarial training (Eq. (9.5)) with adversarial steps of the FGSM and PGD type [Mad+17].

The training curves for the robust WideResnet models on CIFAR-10 are plotted in Fig. 9.5. Robust models adversarially trained with weaker attackers such as uSGD and FGSM are relatively vulnerable to universal perturbations, while robust models from PGD [Mad+17] and uFGSM can resist universal perturbations. We apply PGD (using the sign of the gradient) and ADAM in Algorithm 7 to generate universal perturbations for these robust models, and show such perturbations in Fig. 9.6. Comparing Fig. 9.6 (a,b,c,d) with Fig. 9.6 (e,f,g,h), we see that universal perturbations generated by PGD and ADAM are different but have similar patterns. Universal perturbations generated for weaker robust models have more geometric textures, as shown in Fig. 9.6 (a,d,e,h).

We apply the strongest attack to the validation images of the natural model and various universal adversarially trained models using different update steps. The result are summarized in Table 9.1. Our models become very robust against universal perturbations and have higher accuracies on natural validation examples compared to per-instance adversarial trained models. Note that the PGD trained model is trained on a 7-step per-instance adversary and requires about $4\times$ more computation than ours.

		Validation Accuracy on	
		UnivPert	Natural
(Robust) models trained with	Natural	9.2%	95.2%
	FGSM	51.0%	91.42%
	PGD	86.1%	87.25%
	uADAM (ours)	91.6%	**94.28%**
	uFGSM (ours)	**91.8%**	93.50%

Table 9.1: Validation accuracy of hardened WideResnet models trained on CIFAR-10. Note that Madry's PGD training is significantly slower than the other training methods.

9.4.2 Universal adversarial training for free!

As shown in Table 9.1, our proposed algorithm 7 was able to harden the CIFAR-10 classification network. This comes at the cost of doubling the training time. Adversarial training in general should have some cost since it requires the generation or update of the adversarial perturbation of the mini-batch *before* each minimization step on the network's parameters. However, since universal perturbations are approximately image-agnostic, results should be fairly invariant to the order of updates. For this reason, we propose to compute the image gradient needed for the perturbation update during the same backward pass used to compute the parameter gradients. This results in a simultaneous update for network weights and the universal perturbation in Algorithm 8, which backprops only once per iteration and produces approximately universally robust models at almost no cost in comparison to natural training. The "free universal adversarial trained" model of CIFAR-10 is 86.1% robust against universal perturbations and has 93.5% accuracy on the clean validation examples. When compared to the non-free version in Table 9.1, the robustness has only slightly decreased. However, the training time is

Algorithm 8 Simultaneous stochastic gradient method for adversarial training against universal perturbation

Input: Training samples X, perturbation bound ϵ, learning rate τ, momentum μ

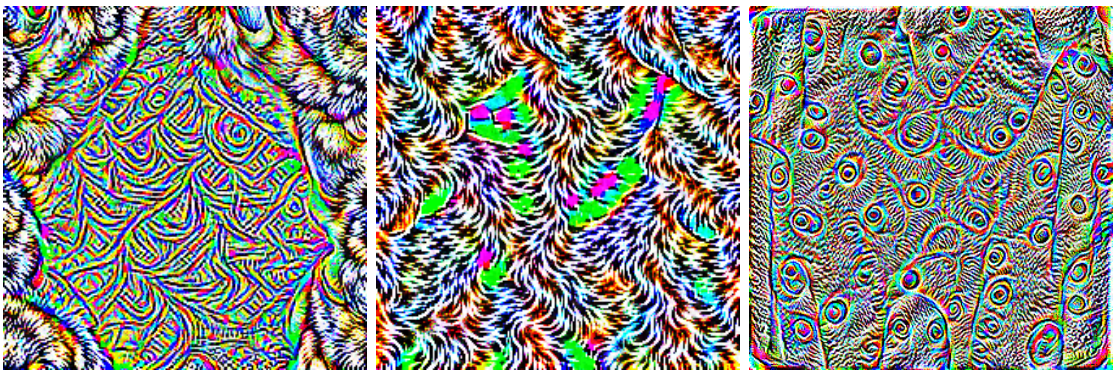
```
Initialize w, δ
for epoch = 1 ... N_ep do
  for minibatch B ⊂ X do
    Compute gradient of loss with respect to w and δ
      d_w ← E_{x∈B}[∇_w l(w, x + δ)]
      d_δ ← E_{x∈B}[∇_δ l(w, x + δ)]
    Update w with momentum stochastic gradient
      g_w ← μ g_w − d_w
      w ← w + τ g_w
    Update δ with stochastic gradient ascent
      δ ← δ + ε sign(d_δ)
    Project δ to ℓ_p ball
  end for
end for
```

cut by half. This is a huge improvement in efficiency, in particular for large datasets like ImageNet with long training times.

9.5 Universal perturbations for ImageNet

To validate the performance of our proposed optimization on different architectures and more complex datasets, we apply Algorithm 6 to various popular architectures designed for classification on the ImageNet dataset [Rus+15]. We compare our method of universal perturbation generation with the current state-of-the-art method, Iterative DeepFool (iDeepFool for short) [MD+17b]. We use the authors' code to run the iDeepFool attack on these classification networks. For fair comparison, we execute both our method and iDeepFool on the exact same 5000 training data points and terminate both methods after 10 epochs. We use $\epsilon = 10$ for ℓ_∞ constraint following [MD+17b], use a step-size of 1.0 for our method, and use sug-

gested parameters for iDeepFool. Similar conclusions could be drawn when we use ℓ_2 bounded attacks. We independently execute iDeepFool since we are interested in the accuracy of the classifier on attacked images – a metric not reported in their paper [1].



(a) InceptionV1 (b) VGG16 (c) InceptionV3

Figure 9.7: Universal perturbations generated using our Algorithm 6 for different network architectures on ImageNet. Visually, these perturbations which are for naturally trained models are structured.

9.5.1 Benefits of the proposed method

We compare the performance of our stochastic gradient method for Eq. (9.2) and the iDeepFool method for Eq. (9.1) in [MD+17b]. We generate universal perturbations for Inception [Sze+16] and VGG [SZ14] networks trained on ImageNet [Rus+15], and report the top-1 accuracy in Table 9.2. Universal perturbations generated by both iDeepFool and our method can fool networks and degrade the classification accuracy. Universal perturbations generated for the training samples generalize well and cause the accuracy of the validation samples to drop. However,

[1] They report "fooling ratio" which is the ratio of examples who's label prediction changes after applying the universal perturbation. This has become an uncommon metric since the fooling ratio can increase if the universal perturbation causes an example that was originally miss-classified to become correctly classified.

		InceptionV1	VGG16
Natural	Train	76.9%	81.4 %
	Val	69.7%	70.9%
iDeepFool	Train	43.5%	39.5%
	Val	40.7%	36.0%
Ours	Train	**17.2%**	**23.1%**
	Val	**19.8%**	**22.5%**
iDeepFool time (s)		9856	6076
our time (s)		**482**	**953**

Table 9.2: Top-1 accuracy on ImageNet for natural images, and adversarial images with universal perturbation.

when given a fixed computation budget such as number of passes on the training data (i.e., epochs), our method outperforms iDeepFool by a large margin. Our stochastic gradient method generates the universal perturbations at a much faster pace than iDeepFool. About 20× faster on InceptionV1 and 6× on VGG16 (13× on average).

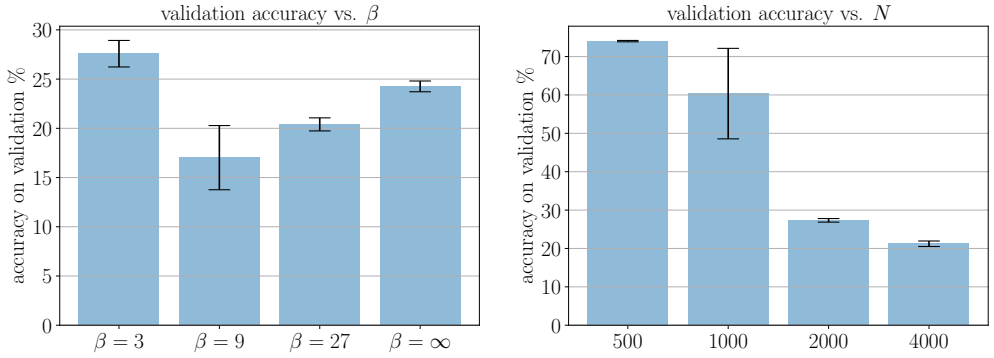
After verifying the effectiveness and efficiency of our proposed stochastic gradient method[2], we use our Algorithm 6 to generate universal perturbations for more advanced architectures such as ResNet-V1 152 [He+16] and Inception-V3 [Sze+16] (and for other experiments in the remaining sections). Our attacks degrade the validation accuracy of ResNet-V1 152 and Inception-V3 from 76.8% and 78% to 16.4% and 20.1%, respectively. The final universal perturbations used for the results presented in this section are illustrated in Fig. 9.7.

[2]Unless otherwise specified, we use the sign-of-gradient PGD for our stochastic gradient optimizer in Algorithm 6.

9.5.2 The effect of clipping

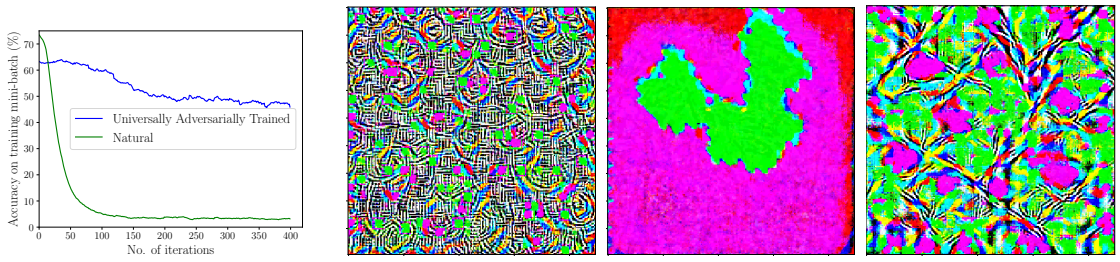
In this section we analyze the effect of the "clipping" loss parameter β in Eq. (9.2). For this purpose, similar to our other ablation experiments, we generate universal perturbations by solving Eq. (9.2) using PGD for Inception-V3 on ImageNet.

Since the results and performance could slightly vary with different random initializations, we run each experiment with 5 random subsets of training data. The accuracy reported is the classification accuracy on the entire validation set of ImageNet after adding the universal perturbation. The results are summarized in Fig. 9.8a. The results showcase the value of our proposed loss function for finding universal adversarial perturbations.



(a) Attack performance varies with clipping parameter β in Eq. (9.2). Attacking Inception-V3 (with natural validation accuracy 78%) is more successful with clipping ($\beta = 9$) than without clipping ($\beta = \infty$).

(b) The attack performance significantly improves when the number of training points is larger than the number of classes. For reference, Inception-V3's top-1 accuracy is 78%. Using only a small fraction of the training-data (4,000 / 1,281,167) is enough to degrade the validation and train accuracy to around 20%.



(a) Generating universal perturbations (b) Natural AlexNet (c) Universal Training (d) Free Universal Training

Figure 9.9: Training universal perturbation can fool naturally trained AlexNet on ImageNet, but fails to fool our robust AlexNets. We smoothed the curves in (a) for better visualization. The universal perturbations generated for the universal adversarial trained AlexNets on ImageNet have little geometric structure compared to that of the naturally trained network. (b) Universal perturbation of natural model. The accuracy of the validation images + noise is only **3.9**% (c) Perturbation for our universally trained model using Algorithm 7. The accuracy of the validation images + noise for our robust model is **42.0%**. (d) Perturbation for the model trained with our free universal training variant (Algorithm 8). The accuracy of the validation images + noise is **28.3**%. While the universal noise for the free variant of universal adversarial training has some structure compared to the non-free variant, when compared to that of the natural model (b), it is structure-less.

9.5.3 How much training data does the attack need?

As in [MD+17b], we analyze how the number of training points ($|X|$) affects the strength of universal perturbations in Fig. 9.8b. In particular, we build δ using varying amounts of training data. For each experiment, we report the accuracy on the entire validation set after we add the perturbation δ. We consider four cases for $|X|$: 500, 1000, 2000, and 4000 [3].

[3]The number of epochs (N_{ep} in Algorithm 6) was 100 epochs for 500 data samples, 40 for 1000 and 2000 samples, and 10 for 4000 samples.

9.6 Universal adversarial training on ImageNet

In this section, we analyze our robust models that are universal adversarially trained by solving the min-max problem (Section 9.4) using Algorithm 7. We use $\epsilon = 10$ for ImageNet following [MD+17b].

Since our universal adversarial training algorithm (Algorithm 7) is cheap, it scales to large datasets such as ImageNet. To illustrate this, we train an AlexNet model on ImageNet. We use the natural training hyper-parameters for universal adversarially training our AlexNet model. Also, we separately use our "free universal training" algorithm to train a robust AlexNet with no overhead cost. We then attack the natural, universally trained, and no-cost universally trained versions of AlexNet using universal attacks.

As seen in Fig. 9.9 (a), the AlexNet trained using our universal adversarial training algorithm (Algorithm 7) is robust against universal attacks generated using both Algorithm 5 and Algorithm 6. The naturally trained AlexNet is susceptible to universal attacks. The final attacks generated for the robust and natural models are presented in Fig. 9.9 (b,c,d). The universal perturbation generated for the robust AlexNet model has little structure compared to the universal perturbation built for the naturally trained AlexNet. This is similar to the trend we observed in Fig. 9.3 and Fig. 9.6 for the WideResnet models trained on CIFAR-10.

The accuracy of the universal perturbations on the validation examples are summarized in Table 9.3. Similar to CIFAR-10, the free version of universal adversarial training is robust but not as robust as the main method.

Training	Evaluated Against			
	Natural Images		Universal Attack	
	Top-1	Top-5	Top-1	Top-5
Natural	56.4%	79.0 %	3.9 %	9.4 %
Universal	49.5%	72.7%	42.0%	65.8 %
Free Univ.	48.4%	72.4%	28.3%	48.3 %

Table 9.3: Accuracy on ImageNet for natural and robust models.

9.7 Summarization

We proposed using stochastic gradient methods and a "clipped" loss function as an effective universal attack that generates universal perturbations much faster than previous methods. To defend against these universal adversaries, we proposed to train robust models by optimizing a min-max problem using alternating or simultaneous stochastic gradient methods. We show that this is possible using certain universal noise update rules that use "normalized" gradients. The simultaneous stochastic gradient method comes at almost no extra cost compared to natural training and is "free". Due to the relatively cheap computational overhead of our proposed universal adversarial training algorithms, we can easily train robust models for large-scale datasets such as ImageNet.

Chapter 10: Exploiting Adaptive Networks for Robustness

In this chapter, we further exploit the practical robustness of neural networks by using our fast algorithm for adversarial training [Sha+18; Sha+19]. We study the effect of network architectures, and present preliminary results on adapting network to be robust to test samples.

10.1 Introduction

Deep neural networks have achieved impressive performance on many machine learning tasks, which has led to growing interests in deploying these models in practical applications. Many research studies have revealed that models trained on benign examples are susceptible to *adversarial examples*, examples crafted by an adversary to control model behavior at test time [Big+13; Sze+13; Goo+15]. The adversarial noise overlaid on top of the benign examples are small enough to be imperceptible for humans. For a classification task, the most common goal of the adversary is to cause the model to misclassify the adversarial example.

The existence of adversarial examples has raised security concerns for many high-stakes real-world applications such as street sign detection for autonomous vehicles. While initial works stated that adversarial examples built for sign-detection

may not be a real threat since the camera can view the objects from different distances and angles [Lu+17], more recent attacks were proposed for making stronger adversarial examples that are invariant to various transformations by optimizing over the expected value of a set of pre-defined transformations [Ath+17]. These security concerns and threats have guided researchers to create models that are both accurate in prediction and robust to attacks.

Various methods have been proposed for defending against adversarial examples. One popular approach is to detect and reject or project adversarial examples [Ma+18; MC17; Xu+17a; Lam+18; Sam+19], which is less effective when the adversary is aware of the detection method and can adopt accordingly [CW17a]. Another approach is to introduce regularization for training robust models [Cis+17; JG18], which can only increase robustness to a limited level. Athalye et al. [Ath+18] showed that many proposed defenses give a false sense of security by obfuscating the gradients, as accurate gradient information is necessary for optimization based attacks. Athalye et al. [Ath+18] broke these defenses by attacks that build good approximations for the gradients. Among various defense methods, adversarial training [Mad+17; Kan+18; Xie+19] is one of the most common methods for training robust models. In adversarial training, a robust model is trained on adversarial examples that are generated on-the-fly, which is effective but also makes adversarial training expensive.

Robustness and robust models have some interesting properties which have been revealed in recent studies. First, it is argued that there exists trade-off between accuracy and robustness [Tsi+18; Zha+19a; Su+18]. It is difficult to make a

model robust to all samples while maintaining the same level of accuracy. Second, it is difficult to adversarially train robust models which generalize since adversarially robust generalization requires more data [Sch+18] and models with high capacity [Mad+17]. Training high capacity models on large datasets increases the cost of adversarially training robust models. Third, while adversarial training is expensive, it is shown that adversarially trained models learn feature representations that align well with human perception [Tsi+18]. These feature embeddings can produce clean inter-class interpolations similar to generative models in Generative Adversarial Networks (GANs) [Goo+14a].

Recently, conditional normalization, built upon instance normalization [Uly+16a] or batch normalization [IS15], has been successful in generative models [Kar+19] and style transfer [HB17]. Conditional normalization can be seen as an adaptive network that shifts the statistics of a layers activations by applying network parameters conditioned on the latent factors such as style and classes [DV+17; Dum+17]. Inspired by these studies, we propose to exploit adaptive networks in the adversarial training framework.

We propose building hardened networks by adversarially training *adaptive networks*. To build adaptive networks, we introduce a normalization module conditioned on *inputs* which allows the network to "adapt" itself for different samples. The conditional normalization module includes a meta convolutional network that changes the scale and bias parameter for normalization based on input samples. Conditional normalization is a powerful module that enlarges the representative capacity of networks. Adversarially trained adaptive nets can be potentially more

robust than conventional non-adaptive nets as they can adapt the network to be robust to adversarial attacks on a specific sample instead of all samples.

Our experiments on the CIFAR-10 and CIFAR-100 benchmarks empirically show that our proposed adaptive networks are better than their non-adaptive counterparts. The adaptive networks even outperform larger networks with more parameters both in terms of accuracy on validation examples and robustness. As we were building strong baselines for our experiments, we observed that adversarial training starting from a naturally trained initialized network helps in improving robustness generalization. Adversarially training our adaptive nets even outperform this new strong baseline.

10.2 Related work

Here we provide a brief overview of robustness and normalization layers which are closely related to our proposed adaptive networks. Given that adversarial training plays a critical role in our method, we also provide an overview on adversarial training.

Robustness is commonly measured by computing the accuracy of the model on adversarial examples constructed by gradient-based optimization methods for validation samples. This evaluation method provides an upper-bound on robustness as there is no theoretical guarantee (at least for all classes of problems) that adversarial examples crafted using first-order gradient information are optimal. From a theoretical point of view, finding optimal adversarial examples is difficult. Some

recent works have proposed finding the optimal solution by modeling neural networks as Mixed Integer Programs (MIPs) and solving those MIPs using commercial solvers [Tje+17]. However, finding the optimal solution of an MIP is generally NP-hard. Although recent advancements have been made in their formulations by enforcing some properties on the network [Xia+19], finding the optimal solution is only feasible for small networks and is very time consuming. That is why certified methods in practice provide lower-bounds on the size of perturbation needed for causing misclassification by solving a relaxed version of the problem.

Raghunathan et al. [Rag+18] propose certified defences by including a differentiable certificate as a regularizer. Many studies follow this line of work and propose certified defenses [Wan+18; Won+18; Coh+19]. While from a theoretical point of view certified defenses are interesting, in practice, adversarial training is still the most popular method for hardening networks – leaders of various computer vision defense competitions and benchmarks utilize adversarial training in their approach [Xie+19; Zha+19a; Mad+17; Sha+18].

Adversarial training, in its general form, corresponds to training on the following loss function,

$$\underset{\theta}{\text{minimize}} \sum_{i} \kappa J(x_i, y_i, \theta) + (1-\kappa) J(x_i + \delta_i, y_i, \theta) \tag{10.1}$$

where J is a differentiable surrogate loss used for training the neural network such as the cross-entropy loss, (x_i, y_i) is the i^{th} data-point and its correct label, θ is the nets trainable parameters, κ is a hyper-parameter which controls how much weight should

be given to training on natural examples, and δ_i corresponds to the adversarial perturbation for the i^{th} datapoint. To keep the perturbation unrecognizable to humans, δ_i is usually bounded by some norm. Throughout this paper, we will use the common ℓ_∞-norm bound on δ. Effectively, this adversarial training loss has two terms, one term which trains on natural examples and the second term which trains on adversarial examples. This corresponds to training on batches which have both natural and adversarial examples.

Early adversarial example generation methods required many iterations since their goal was to help an attacker build an adversarial example which has minimum perturbation [Sze+13; MD+16; CW17b]. However, from a defenders perspective, the goal is to train on fast and bounded adversarial examples. With speed in mind, Goodfellow et al. [Goo+15] proposed training on a single-step ℓ_∞ attack called the Fast Gradient Sign Method (FGSM). FGSM computes $\nabla_x J(x, y, \theta)$ and sets $\delta = \epsilon \cdot \text{sign}(\nabla_x J(x, y, \theta))$, where ϵ is the perturbation bound. Later, it was shown that stronger attacks such as BIM [Kur+16a], completely break FGSM adversarially trained models. The BIM attack can be seen as an iterative version of FGSM where during each iteration, the perturbation is updated using an FGSM-type step but with a step-size ϵ_s which is usually smaller than ϵ,

$$\delta^k = \delta^{k-1} + \epsilon_s \cdot \text{sign}(\nabla_\delta J(x + \delta^{k-1}, y, \theta)) \tag{10.2}$$

where δ^k is the perturbation at iteration k of the BIM attack. After every iteration of the BIM attack (equation 10.2), δ^k is clipped such that $\delta^k \in [-\epsilon, \epsilon]$. We refer to

the K-iteration BIM attack as BIM-K.

Adversarial training started blooming when Madry et al. [Mad+17] proposed training on adversarial examples generated using the PGD attack. The PGD attack is the BIM with a random initialization. Through experiments, they showed that the PGD attack is the strongest first-order adversary. This was later verified by Athalye et al. [Ath+18] as well. Consequently, almost all of the successful adversarial trained robust models use the PGD algorithm to generate adversarial examples .

Training on adversarial examples generated using PGD increases the cost of training by a factor of K, where K is the number of iterations of the PGD attack (i.e., number of times we update δ using equation 10.2). While we will use PGD-K attacks for evaluating the robustness of all our models, due to the high computation cost associated with PGD adversarial training, we perform most of our adversarial training using a recently proposed algorithm for speeding up adversarial training [Sha+19].

Normalization layers such as batch normalization [IS15] and instance normalization [Uly+16a] have become important modules in modern neural networks. Normalization layers standardize input to have zero mean and one variance and then shift these statistics using scaling and bias parameters. Zhang et al. [Zha+19b] suggest scaling and bias parameters can be even more important than standardization. Conditional normalization, where scaling and bias are adaptively determined by latent factors, has shown to be powerful in many computer vision tasks including style transfer [HB17; Dum+17] and generative adversarial networks [Kar+19].

10.3 Adaptive Networks

We introduce adaptive networks with conditional normalization modules in this section. Our motivation for adding conditional normalization modules is two-fold. First, by introducing adaptive layers conditioned on inputs, we can "adapt' a trained network to be more robust to individual input sample.

Second, conditional normalization can increase the expressiveness and effective capacity of the network, which has been shown to have a positive effect in improving model robustness. Adversarially trained models with more expressive capacities are more robust than their less expressive alternatives [Mad+17; Sha+19]. At a high level, these conditional normalization modules can be considered as adding multi-branch structures to a network which is known to be effective in improving accuracy on validation examples [Hua+17]. As we will see in the experiments, our normalization module indeed does improve the clean validation accuracy and is more effective[1] than simply widening or concatenating features in practice.

Below, we show how to create an adaptive network by adding conditional normalization modules to the wide residual network (WRN) [ZK16] architecture. We also briefly review the fast adversarial training algorithm we will use to make our adaptive networks robust.

[1]The adversarially trained adaptive nets have higher validation accuracy and robustness compared to networks with more trainable parameters.

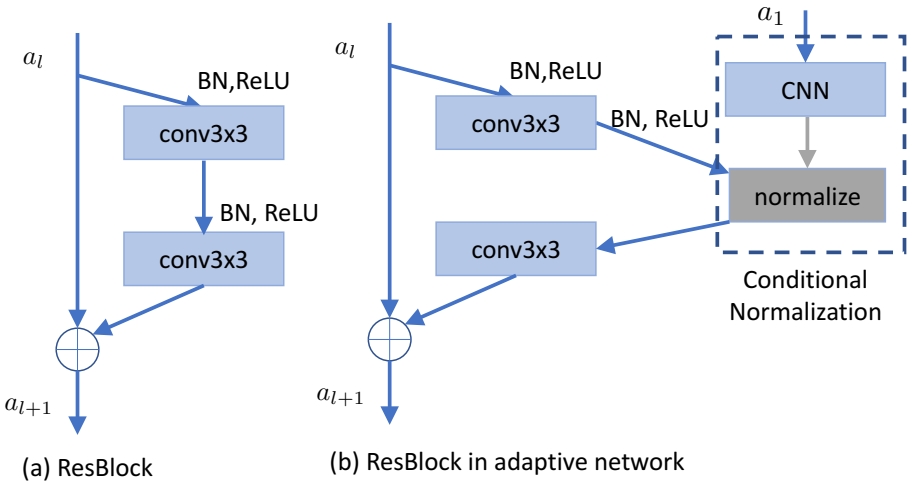


Figure 10.1: Network architecture with adaptive layers.

10.3.1 Network architecture

Let $x \in \mathbb{R}^{N\times C\times H\times W}$ represent the feature maps of a convolutional layer for a minibatch of samples, where N is the batch size, C is the width of the layer (number of channels), and H and W are the feature map's height and width. If x_{nchw} denotes the element at height h, width w of the c^{th} channel from the n^{th} sample, the conditional normalization module transforms the feature maps as,

$$\text{Norm}(x_{nchw}|z) = \nu(z)_{nc}\, x_{nchw} + \mu(z)_{nc}, \tag{10.3}$$

where $\nu(z), \mu(z) \in \mathbb{R}^{N\times C}$ are scale and bias parameters of the normalization module. The network with conditional normalization becomes adaptive to the latent factor z as $\nu(z), \mu(z)$ are outputs of convolutional networks with learnable parameters. Equation 10.3 represents normalization in a general form: when latent factor z is a style image and x is normalized by its mean and variance, equation 10.3 becomes adaptive instance normalization for image style transfer [HB17]; when latent factor z

is latent code like random noise, equation 10.3 becomes the building module for the generator in StyleGAN [Kar+19]. We provide details on how we use input sample as latent factor z as below.

We now add conditional normalization module to wide residual network (WRN) [ZK16] to create adaptive networks for classification. WRN is an derivative of ResNet [He+16] and is one of the state-of-the-art architectures used for image classification. WRN is a stack of residual blocks (Fig. 10.1(a)). To specify WRNs, we follow [ZK16] and denote the architecture as WRN-β-α, where β represents the depth and α represents the widening factor of the network.

The WRN architecture for the CIFAR-10 and CIFAR-100 datasets we use in this paper consists of a stack of three groups of residual blocks. There is a downsampling layer between two groups, and the number of channels (width of a convolutional layer) is doubled after downsampling. In the three groups, the width of convolutional layers is $\{16\alpha, 32\alpha, 64\alpha\}$, respectively. Each group contains β_r residual blocks, and each residual block contains two 3×3 convolutional layers equipped with ReLU activation and batch normalization. There is a 3×3 convolutional layer with 16 channels before the three groups of residual blocks. And there is a global average pooling, a fully-connected layer and a softmax layer after the three groups. The depth of WRN is $\beta = 6\beta_r + 4$.

We add conditional normalization for the first residual block of each of the three groups. The normalization module is applied between the two convolutional layers in a block, as shown in Fig. 10.1(b). The inputs to the conditional normalization module are the feature maps produced by the first convolutional layer. Our

conditional normalization module consists of a three layer convolutional network: two 3×3 convolutional layers with 16α, and one 1×1 convolutioinal layer to match the dimension of the three different residual blocks, $2 \times \{16\alpha, 32\alpha, 64\alpha\}$, respectively. We use average pooling as the last layer to get $\nu(z), \mu(z)$ for equation 10.3. Our adaptive network is only slightly larger than the original WRN, and becomes more robust when adversarially trained, as shown in Section 10.3.2.

10.3.2 Adversarial training

Well-known robust networks on MNIST and CIFAR-10 were adversarially trained by Madry et al. [Mad+17] by setting $\kappa = 0$ in equation 10.1. When $\kappa = 0$, training is only done on adversarial examples. Training just on adversarial examples is justified from a robust optimization framework. In this framework, adversarial training is modeled as a two-player constant sum game between the adversary which is in charge of the perturbation δ and the minimizer which controls the network's parameters θ. Formally, the adversarial training they propose and the one we use in this paper is based on the following saddle point formulation,

$$\underset{\theta}{\text{minimize}} \ \underset{\delta_i}{\text{maximize}} \sum_i J(x_i + \delta_i, y_i, \theta) \qquad \text{subject to:} \quad \|\delta_i\|_\infty \leq \epsilon \quad \forall i \qquad (10.4)$$

Madry et al. [Mad+17] solved the optimization problem in equation 10.4 in an alternating fashion. Before each minimization step on the network parameters θ, they compute δ using a PGD-K attack on the fly. Every perturbation update step of the PGD-K attack (equation 10.2) requires computing $\nabla_\delta J(x_j + \delta_j^{k-1}, y_j, \theta^j)$, where

δ_j^{k-1} are the adversarial perturbations of the j$^{\text{th}}$ mini-batch after the previous $k-1$ times δ update step, and θ^j represents network parameters at the j$^{\text{th}}$ minimization iteration. To compute $\nabla_\delta J(x_i + \delta_i^{k-1}, y_i, \theta)$, required for every step of PGD-K, we need to do a complete forward and backward pass on the network. As a result, every minimization iteration of PGD adversarial training costs $(K+1)\times$ every minimization iteration of natural training. For CIFAR-10 a typical value used for K is 7 [Mad+17].

To speed up training robust models by solving the optimization problem expressed in equation 10.4, we adopt a fast adversarial training algorithm recently proposed by Shafahi et al. [Sha+19]. Shafahi et al. [Sha+19] showed that they can achieve comparable robustness to PGD adversarial training [Mad+17] on the datasets of our interest (CIFAR-10 and CIFAR-100) while being roughly $(K+1)$-times faster. Where K is the number of steps the PGD algorithm.

The fast algorithm (Free-m) has a perturbation parameter δ_b of shape $N \times C \times H \times W$ which is updated once during every minimization iteration. To accelerate robust training, Free-m applies simultaneous updates for the network parameters θ and perturbation δ, which makes its computation cost almost the same as natural training. During the j$^{\text{th}}$ minimization iteration, both $\nabla_\delta J$ and $\nabla_\theta J$ are computed for the current mini-batch (x_j, y_j) and network parameters θ^j,

$$\nabla_\theta J = \mathbb{E}_{\{(x_j, y_j)\}}[\nabla_\theta \, J(x_j + \delta_b^j, y_j, \theta^j)]$$

$$\nabla_\delta J = \nabla_\delta \, J(x_j + \delta_b^j, y_j, \theta^j)]$$

Then θ and δ are updated as,

$$\theta^{j+1} = \theta^j - \tau \nabla_\theta J$$

$$\delta_b^{j+1} = \text{clip}(\delta_b^j + \epsilon_s \cdot \text{sign}(\nabla_x J), -\epsilon, \epsilon).$$

In Free-m, each mini-batch is replayed m times. For example, if $m = 2$, we move on to the next mini-batch every m steps and therefore the data for the first two iterations would be the same (i.e., $(x_1, y_1) = (x_2, y_2)$). Since we train on the same mini-batch m-times in a row, the hyper-parameter m is more-or-less analogous to the number of iterations of the PGD training algorithm K. We use the same number of minibatch updates for Free-m adversarial training and natural training on clean images, i.e., we train Free-m for $1/m$ number of epochs in total.

In this section, we train robust models on CIFAR-10 and CIFAR-100. In all the experiments, we train WRN without dropout for 120 epochs and with minibatch size 256. We start with learning rate 0.1 and decrease the learning rate by a factor of 10 at epochs 60 and 90. We use weight decay 1e-4 and momentum 0.9. For evaluating the robustness of the models, we attack them with PGD-K attacks. For the PGD attacks, we use $\epsilon_s = 2$ and $\epsilon = 8$, and vary the number of attack iterations K.

10.3.3 Quantitative evaluation on CIFAR-10 and CIFAR-100

We summarize our quantitative evaluation on CIFAR-10 and CIFAR-100 in Table 10.1, Table 10.2, and Table 10.3. In Table 10.1 and Table 10.2, unless oth-

(Robust) model	Evaluated Against			#Parameter (million)
	Natural	PGD-20	PGD-100	
Natural	**94.10%**	0.00%	0.00%	5.85
PGD-7-small [Mad+17]	83.84%	40.03%	39.38%	5.85
Free-10-small [Sha+19]	81.04%	40.56%	40.03%	5.85
Free-10-adaptive-small	85.00%	43.16%	42.68%	6.05
Free-10 [Sha+19]	77.75%	45.10%	44.77%	5.85
Free-10-WRN-28-5	77.81%	45.99%	45.77%	9.13
Free-10-init	80.60%	46.88%	46.67%	5.85
Free-10-adaptive	80.99%	**48.09%**	**47.87%**	6.05

Table 10.1: Performance of (robust) CIFAR-10 models. We inject adaptive layers in WRN-28-4, and compare with WRN-28-4 and WRN-28-5 with more parameters.

(Robust) model	Evaluated Against			#Parameter (million)
	Natural	PGD-20	PGD-100	
Natural	**74.84%**	0.00%	0.00%	5.87
PGD-7-small [Mad+17]	57.18%	18.38%	18.13%	5.87
Free-10-small [Sha+19]	54.18%	19.21%	18.98%	5.87
Free-10-adaptive-small	61.19%	21.95%	21.68%	6.07
Free-10 [Sha+19]	50.52%	23.08%	23.02%	5.87
Free-10-WRN-28-5	51.02%	23.12%	23.03%	9.16
Free-10-init	55.93%	24.86%	24.61%	5.87
Free-10-adaptive	57.26%	**25.86%**	**25.69%**	6.07

Table 10.2: Performance of (robust) CIFAR-100 models. We inject adaptive layers in WRN-28-4, and compare with WRN-28-4 and WRN-28-5 with more parameters.

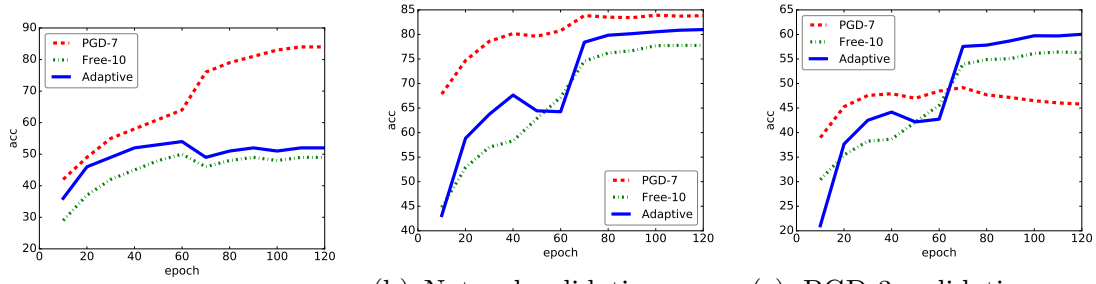
erwise explicitly specified through the name of the model, the architecture used for producing these results is WRN-28-4. We report validation accuracy on natural images and adversarial images generated using PGD attacks with $K = 20$ iterations and $K = 100$ iterations. We also compare our method with adversarially trained robust models following [Mad+17] and [Sha+19]. Note that the PGD-7 adversarially trained model [Mad+17] requires $\approx 7\times$ more time than natural training on clean images, while the Free-10 models [Sha+19] have similar computation cost as natural training. Models with the suffix "small" are adversarially trained using a step-size of $\epsilon_s = 2$. The adversarially trained models without the small suffix are

(Robust) model	Evaluated Against			#Parameter (million)
	Natural	PGD-20	PGD-100	
Natural	**94.76%**	0.00%	0.00%	46.16
PGD-7 from [Mad+17]	87.3%	45.8%	45.3%	45.90
Free-8 from [Sha+19]	85.96%	46.82%	46.19%	45.90
Free-10 [Sha+19]	79.45%	48.03%	47.9 %	46.16
Free-10-init	84.03%	50.23%	49.93%	46.16
Free-10-adaptive	84.39%	**50.93%**	**50.68%**	47.28

Table 10.3: Performance of (robust) CIFAR-10 WRN-34-10 models. We directly compare with previously reported results in [Mad+17; Sha+19].

trained with a step-size $\epsilon_s = 6$.

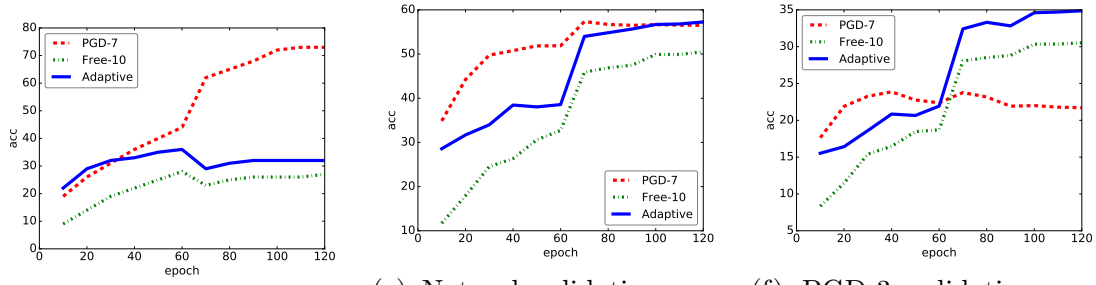
We first evaluate robust models trained with step-size $\epsilon_s = 2$ for perturbation updates following [Mad+17] (rows 2-4 in Table 10.1 and Table 10.2). We can train a robust WRN-28-4 with PGD-7-small [Mad+17] that achieves about 40% accuracy under strong PGD attacks. Our alternative adversarial training mechanism, Free-10-small [Sha+19] achieves slightly better robust accuracy under PGD attacks with a drop in natural accuracy on clean validation images. Since Free-10 is significantly faster than PGD adversarial training, we also use it to adversarially train our adaptive networks. Our adaptive network with conditional normalization built off of WRN-28-4 (Free-10-adaptive-small) outperforms the PGD adversarially trained WRN-28-4 (PGD-7-small) and Free-10-small in both natural accuracy and robust accuracy illustrating the advantage of our adaptive networks. Shafahi et al. [Sha+19] reported results based on a larger stepsize for perturbation updates ($\epsilon_s = \epsilon$). As we show in the 5th row of Table 10.1 and Table 10.2, by comparing Free-10 and Free-10-small, we can see that the larger step-size used for training does improve the robustness of free training but again at an additional cost of decreasing natural validation accuracy.



(a) Training accuracy.

(b) Natural validation accuracy.

(c) PGD-3 validation accuracy.



(d) Training accuracy.

(e) Natural validation accuracy.

(f) PGD-3 validation accuracy.

Figure 10.2: Training curves for robust models for (top) CIFAR-10 and (bottom) CIFAR-100: (left) accuracy on adversarial training samples; (middle) accuracy on clean validation samples; (right) accuracy on PGD-3 validation samples.

We provide two more stronger baselines: adversarially train a larger model WRN-28-5 (row 6), and train WRN-28-4 with a naturally trained model as initialization (row 7). Our Free-10-adaptive-small model had slightly more parameters compared to the adversarially trained PGD-7-small and Free-10-small models. The baseline with more capacity was added to ensure that the superiority of our adaptive network is not solely due to having (slightly) more parameters. Our adaptive network is slightly larger than the non-adaptive WRN-28-4, and is much smaller than WRN-28-5. A good initialization surprisingly helps both natural accuracy and robust accuracy. Our adaptive network outperforms the best strong baseline for both natural accuracy and robust accuracy.

In Table 10.3, we report results on a larger network WRN-34-10, which is

widely used for the CIFAR-10 benchmark. Besides baseline models trained by Free-10 and Free-10-init (Free-10 with good initialization), we also directly compare with the accuracy values reported in the literature on an almost identical network with slightly different training settings in [Mad+17] and [Sha+19]. Like before, our adaptive network outperforms Free-10 and Free-10-init on both natural accuracy and robust accuracy. Comparing with previously reported results, our adaptive network achieves better robust accuracy and only slightly worse natural accuracy.

10.3.4 Training curves and qualitative analysis

We plot the training and validation accuracy of the Free-10, Free-10-adaptive, and PGD-7 adversarially trained (PGD-7) models after each epoch in Fig. 10.2. The training accuracies are computed for the adversarial examples they are being trained on and do not correspond to the natural training accuracy. They can be thought of as robustness on training examples. In Figs. 10.2a and 10.2d, the PGD-7 model fits the adversarial examples built for the training samples to a rather high accuracy, while Free-10 seems to never overfit to the training-set adversarial training samples. The training accuracy of Free-10 [Sha+19] is quite close to the final adversarial validation accuracy in Figs. 10.2c and 10.2f. The natural validation accuracy of PGD-7 increases faster than Free-10 at the beginning, while the accuracy at the end of training become close, as shown in Figs. 10.2b and 10.2e. Free-10 consistently improves robust accuracy against adversarial validation samples, while PGD-7 seems to saturate after the fast increase at the beginning (see Figs. 10.2c

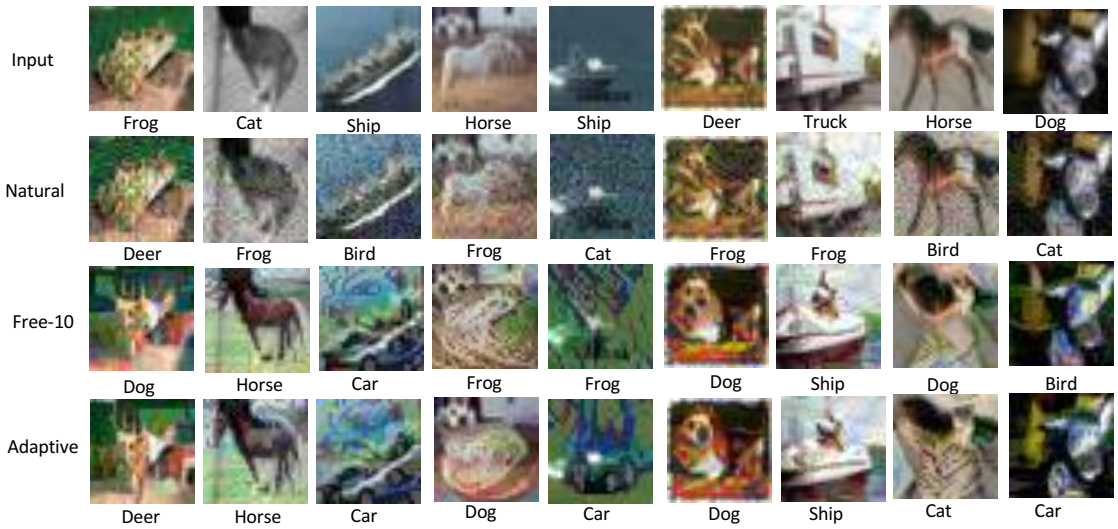


Figure 10.3: Visualization of adversarial examples generated for natural and robust WRN-34-10 for CIFAR-10 with large $\epsilon = 30$ following [Tsi+18]. The large ϵ adversarial examples generated for robust models align well with human perception.

and 10.2f). Our adaptive network (blue curve) always has higher natural and robust validation accuracy than the non-adaptive WRN-28-4 models except for a short range around epoch 60 in Figs. 10.2b and 10.2c, where the accuracy of the adaptive network decreases. Tuning the learning rate could potentially prevent this decrease and further boost the performance of adaptive networks.

Tsipras et al. [Tsi+18] presented an interesting side effect of robust models: largely perturbed adversarial examples for adversarially robust models align with human perception. That is, they "look" like the class which they are getting misclassified to. We use PGD-50 to generate adversarial images with large perturbations ($\epsilon = 30$). The generated images for our adversarially trained adaptive nets have characteristics that align well with human perception.

10.4 Summarization

Inspired by recent research in conditional normalization [HB17; Kar+19] and properties of robustness [Mad+17; Tsi+18; Sch+18], we introduced an adaptive normalization module conditioned on inputs for boosting the robustness of networks. Our adaptive networks combined with a fast adversarial training algorithm [Sha+19], can effectively train robust models that outperform their non-adaptive parallels and also non-adaptive networks with more parameters. Our experiments on CIFAR-10 and CIFAR-100 benchmark and WRN networks verified the effectiveness and efficiency of adaptive networks.

Chapter 11: Conclusion and Discussion

We studied optimization problems from the training procedure of data driven machine learning models, with a particular focus on minimax problems and their applications. Specifically, we study constrained problem, which is popular in classical convex and nonconvex optimization regime and linear statistical learning problems; then adversarial networks, which are widely used for (conditional) generative models and image processing tasks; and robust models defend against adversarial attacks.

In Part I, we first study minimax problem of the Lagrangian saddle point problem for constrained problems. Many objective can be formulated in a general form of summation of two functions with a linear constraint, and a versatile procedure named ADMM can be applied to solve the equivalent Lagrangian of such problem. Typical applications include sparse regularized linear regression, support vector machine classifier, total variational image denoising, phase retrieval, consensus problem in distributed computing, as introduced in Chapter 2. We focus on adaptively choosing free hyperparameter in ADMM (and its variants), and provide both theoretical and empirical analysis. In Chapter 3, we provide moderate conditions to theoretically guarantee the $O(1/k)$ convergence rate of ADMM with adaptive penalty parameters. In Chapter 4, we propose adaptive ADMM (AADMM) with spectral

stepsize selection, which is a fully automated solve that can be easily used by non-experts to solve constrained problems. AADMM achieves fast practical convergence with a theoretical convergence guarantee. In Chapter 5, we further exploit the key idea of the adaptive schema and propose various variants of AADMM: ARADMM with faster practical convergence, especially for difficult problems like total variational image denoising; ACADMM tackles the difficulty of stepsize estimation in consensus problem for large-scale distributed optimization; AMADMM for splitting optimization problem into multi-blocks; and applying AADMM to nonconvex problems, in which we care about not only the convergence speed but also the quality of solution. We verified the performance of our solver on various optimization problems and benchmark datasets.

In Part II, we study one of the most popular minimax problem in recent years, which is the training of generative adversarial networks (GANs). In Chapter 6, we present prediction step to stabilize stochastic alternating gradient method, inspired by classical convex-concave saddle point optimization. In Chapter 7, we apply adversarial networks to image processing tasks, image style transfer and image dehazing. We adversarially train a single feed-forward network to learn from multi-domain artistic images for arbitrary style transfer; we propose a simple yet efficient network that is difficult to outperform by complicated dehazing methods, and apply GAN framework to train without paired images that are difficult to harness. In Chapter 8, we use adversarial networks for an unconventional application, network acceleration, and provide a systematical study on how to choose a proper network for acceleration.

In Part III, we study the training of robust models against adversarial attacks. Machine learning models may make wrong prediction for adversarial examples that can be generated by adding small perturbations to test samples by gradient based method. In Chapter 9, we study fast algorithm for both attack and defense of universal perturbations that can be added to a set of samples to generate adversarial examples. In Chapter 10, we study adaptive network that can boost the robustness of networks by fast adversarial training.

The recent success of machine learning models benefits from the large-scale data harnessed from internet and labeled through crowd sourcing, the advanced computing power such as GPUs, and models like neural networks to learn strong representation. Besides the powerful back propagation and methods to enhance backprop like batch normalization and residual connections, the progress of deep neural networks often relies on simply more data, larger models and training longer. The *limited* success of machine learning models has mostly been on perception tasks which depend on memorizing representations. There remains many challenges and questions in this exciting research area. It is not necessary for intelligent machines to mimic the behavior and mechanism of humans, but is momerizing and fitting large-scale data enough? Even for fitting large scale data, is neural networks, which are more advanced in practice and dominate a lot of applications with sate-of-the-art performance while rely on many mysterious practical experience to tune, the answer? As it is difficult and expensive to harness data with labels, how to efficiently utilize unsupervised data on the web, and how to protect users' privacy in various applications when data are used? If neural network is the right choice for learning,

how to design architecture for joint optimization of performance and training?

Specifically, the optimization problem of training neural networks relies on efficient back prop, which is one of the reasons for the success of batch normalization and residual connections. We exploit ADMM [Tay+16] as an alternative to SGD for training neural networks, which achieves limited success. Is backprop necessary and how to further improve the efficiency of backprop? The training of networks on large-scale data needs a lot of computation power, and the current practical solution is to simply distribute the computation with data parallelism of large minibatchs. Large minibatch training quickly reaches bottleneck of scalability and can only use limited distributed computing power due to underfitting, generalization and communication. How to design more powerful distributed optimization method for large scale training? The computation resources are not only limited but also expensive, how to design algorithm in a low resource setting? The simple SGD method remains powerful in training neural networks. "Adaptive" methods like ADAM and K-FAC got mixed results, and have more hyper-parameters to tune. Learned meta-optimizers incorporate the design of conventional optimizer as features, and cannot generalize well to unseen data and models. How to develop fast and automated optimizer for large-scale, high-dimensional, nonconvex and nonsmooth problem of neural network training?

More specifically, the minimax problem, especially for training neural networks, is more difficult. The optimization of GAN is still difficult and mysterious. Adversarial training for robust models is another popular minimax problem for neural networks. GAN and adversarial training are related but quite different. It seems

that GAN targets on saddle point solution while adversarial training is more like robust optimization because of the bounded constraint. Another interesting observation is that our fast adversarial training algorithm never overfits training data, but still achieves strong robustness comparing to conventional adversarial training [Xu+19b]. We will continue studying the connection between GAN and adversarial training, and seeking inspiration from classical optimization methods.

With the growing interest in applying machine learning in various practical applications, more and more challenging problems appear. While designing efficient algorithms to tackle domain specific problems is a nontrivial task, both fundamental study and empirical practice are important, especially for the large-scale optimization problem in machine learning.

Bibliography

[Akh+18] Naveed Akhtar, Jian Liu, and Ajmal Mian. "Defense against Universal Adversarial Perturbations". In: *CVPR* (2018).

[Alm+18] Amjad Almahairi, Sai Rajeswar, Alessandro Sordoni, Philip Bachman, and Aaron Courville. "Augmented CycleGAN: Learning Many-to-Many Mappings from Unpaired Data". In: *ICML* (2018).

[Anc+16] Cosmin Ancuti, Codruta O Ancuti, and Christophe De Vleeschouwer. "D-HAZY: A dataset to evaluate quantitatively dehazing algorithms". In: *ICIP*. IEEE. 2016, pp. 2226–2230.

[Anc+18a] Codruta O Ancuti, Cosmin Ancuti, Radu Timofte, and Christophe De Vleeschouwer. "I-HAZE: a dehazing benchmark with real hazy and haze-free indoor images". In: *arXiv preprint arXiv:1804.05091* (2018).

[Anc+18b] Codruta O Ancuti, Cosmin Ancuti, Radu Timofte, and Christophe De Vleeschouwer. "O-HAZE: a dehazing benchmark with real hazy and haze-free outdoor images". In: *arXiv preprint arXiv:1804.05101* (2018).

[Arj+17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein GAN". In: *ICML* (2017).

[Ath+17] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. "Synthesizing robust adversarial examples". In: *arXiv preprint arXiv:1707.07397* (2017).

[Ath+18] Anish Athalye, Nicholas Carlini, and David Wagner. "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples". In: *ICML* (2018).

[Aza+18] Samaneh Azadi, Matthew Fisher, Vladimir Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. "Multi-Content GAN for Few-Shot Font Style Transfer". In: *CVPR* (2018).

[BA+16] Dana Berman, Shai Avidan, et al. "Non-local image dehazing". In: *CVPR*. 2016.

[BB88] Jonathan Barzilai and Jonathan Borwein. "Two-point step size gradient methods". In: *IMA J. Num. Analysis* 8 (1988), pp. 141–148.

[BC14] Jimmy Ba and Rich Caruana. "Do deep nets really need to be deep?" In: *NIPS*. 2014, pp. 2654–2662.

[Bel+18] Vasileios Belagiannis, Azade Farshad, and Fabio Galasso. "Adversarial Network Compression". In: *arXiv preprint arXiv:1803.10750* (2018).

[BF18] Shumeet Baluja and Ian Fischer. "Adversarial transformation networks: Learning to generate adversarial examples". In: *AAAI* (2018).

[Big+13] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. "Evasion attacks against machine learning at test time". In: *ECML-PKDD*. Springer. 2013, pp. 387–402.

[BM03] Samuel Burer and Renato DC Monteiro. "A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization". In: *Mathematical Programming* 95.2 (2003), pp. 329–357.

[Bot+18] Léon Bottou, Frank E Curtis, and Jorge Nocedal. "Optimization methods for large-scale machine learning". In: *SIAM Review* 60.2 (2018), pp. 223–311.

[Bou+13] Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. "Sparse iterative closest point". In: *Computer graphics forum*. Vol. 32. 5. Wiley Online Library. 2013, pp. 113–123.

[Boy+11] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. "Distributed optimization and statistical learning via the alternating direction method of multipliers". In: *Found. and Trends in Mach. Learning* 3 (2011), pp. 1–122.

[Bro+17] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. "Adversarial patch". In: *arXiv preprint arXiv:1712.09665* (2017).

[Buc+06] Cristian Bucilu, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression". In: *KDD*. ACM. 2006, pp. 535–541.

[Byr+95] Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. "A limited memory algorithm for bound constrained optimization". In: *SIAM Journal on Scientific Computing* 16.5 (1995), pp. 1190–1208.

[Cai+14] Xingju Cai, Deren Han, and Xiaoming Yuan. "The direct extension of ADMM for three-block separable convex minimization models is convergent when one function is strongly convex". In: *Optimization Online* 4 (2014).

[Cai+16] Bolun Cai, Xiangmin Xu, Kui Jia, Chunmei Qing, and Dacheng Tao. "Dehazenet: An end-to-end system for single image haze removal". In: *IEEE TIP* 25.11 (2016), pp. 5187–5198.

[Can+15] Emmanuel J Candes, Xiaodong Li, and Mahdi Soltanolkotabi. "Phase retrieval via Wirtinger flow: Theory and algorithms". In: *IEEE Transactions on Information Theory* 61.4 (2015), pp. 1985–2007.

[Cha+16a] Tsung-Hui Chang, Mingyi Hong, Wei-Cheng Liao, and Xiangfeng Wang. "Asynchronous distributed alternating direction method of multipliers: Algorithm and convergence analysis". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 4781–4785.

[Cha+16b] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, and Yann LeCun. "Entropy-sgd: Biasing gradient descent into wide valleys". In: *arXiv preprint arXiv:1611.01838* (2016).

[Cha07] Rick Chartrand. "Exact reconstruction of sparse signals via nonconvex minimization". In: *IEEE Signal Processing Letters* 14.10 (2007), pp. 707–710.

[Che+14] Yunmei Chen, Guanghui Lan, and Yuyuan Ouyang. "Optimal primal-dual methods for a class of saddle point problems". In: *SIAM Journal on Optimization* 24.4 (2014), pp. 1779–1814.

[Che+16a] Caihua Chen, Bingsheng He, Yinyu Ye, and Xiaoming Yuan. "The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent". In: *Mathematical Programming* 155.1-2 (2016), pp. 57–79.

[Che+16b] Chen Chen, Minh N Do, and Jue Wang. "Robust image and video dehazing with visual artifact suppression via gradient residual minimization". In: *ECCV*. Springer. 2016, pp. 576–591.

[Che+16c] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. "Infogan: Interpretable representation learning by information maximizing generative adversarial nets". In: *NIPS*. 2016.

[Che+17a] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. "Stylebank: An explicit representation for neural image style transfer". In: *CVPR*. 2017.

[Che+17b] Yuntao Chen, Naiyan Wang, and Zhaoxiang Zhang. "DarkRank: Accelerating Deep Metric Learning via Cross Sample Similarities Transfer". In: *arXiv preprint arXiv:1707.01220* (2017).

[Che+18] Ziang Cheng, Shaodi You, Viorela Ila, and Hongdong Li. "Semantic Single-Image Dehazing". In: *arXiv preprint arXiv:1804.05624* (2018).

[Cho+15] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. "The Loss Surfaces of Multilayer Networks." In: *AISTATS*. 2015.

[Chr+17] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. "A Downsampled Variant of ImageNet as an Alternative to the CIFAR datasets". In: *arXiv preprint arXiv:1707.08819* (2017).

[Cim+14] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi. "Describing Textures in the Wild". In: *CVPR*. 2014.

[Cis+17] Moustapha Cissé, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. "Parseval Networks: Improving Robustness to Adversarial Examples". In: *ICML*. 2017, pp. 854–863.

[CL11] Chih-Chung Chang and Chih-Jen Lin. "LIBSVM: a library for support vector machines". In: *ACM Transactions on Intelligent Systems and Technology (TIST)* 2.3 (2011), p. 27.

[Coh+19] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. "Certified adversarial robustness via randomized smoothing". In: *arXiv preprint arXiv:1902.02918* (2019).

[CP11] Antonin Chambolle and Thomas Pock. "A first-order primal-dual algorithm for convex problems with applications to imaging". In: *Journal of Mathematical Imaging and Vision* 40.1 (2011), pp. 120–145.

[CS16] Tian Qi Chen and Mark Schmidt. "Fast patch-based style transfer of arbitrary style". In: *arXiv preprint arXiv:1612.04337* (2016).

[CW13] Rick Chartrand and Brendt Wohlberg. "A nonconvex ADMM algorithm for group sparsity with sparse groups". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2013, pp. 6009–6013.

[CW17a] Nicholas Carlini and David Wagner. "Adversarial examples are not easily detected: Bypassing ten detection methods". In: *ACM Workshop on Artificial Intelligence and Security*. ACM. 2017, pp. 3–14.

[CW17b] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks". In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 39–57.

[Cyb89] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *MCSS* 2.4 (1989), pp. 303–314.

[Din+17] Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. "Sharp minima can generalize for deep nets". In: *ICML* (2017).

[DL14] Cong Dang and Guanghui Lan. "Randomized first-order methods for saddle point optimization". In: *arXiv preprint arXiv:1409.8625* (2014).

[Don+14] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. "Decaf: A deep convolutional activation feature for generic visual recognition". In: *ICML*. 2014, pp. 647–655.

[Don+17] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Xiaolin Hu, Jianguo Li, and Jun Zhu. "Boosting adversarial attacks with momentum". In: *CVPR* (2017).

[Du+17] Simon S Du, Jianshu Chen, Lihong Li, Lin Xiao, and Dengyong Zhou. "Stochastic Variance Reduction Methods for Policy Evaluation". In: *ICML* (2017).

[Dum+17] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. "A learned representation for artistic style". In: *ICLR* (2017).

[DV+17] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. "Modulating early visual processing by language". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6594–6604.

[DY14] Damek Davis and Wotao Yin. "Faster convergence rates of relaxed Peaceman-Rachford and ADMM under regularity assumptions". In: *arXiv preprint arXiv:1407.5210* (2014).

[DY16] Wei Deng and Wotao Yin. "On the global and linear convergence of the generalized alternating direction method of multipliers". In: *Journal of Scientific Computing* 66.3 (2016), pp. 889–916.

[DZ13] Bin Dong and Yong Zhang. "An efficient algorithm for ℓ_0 minimization in wavelet frame based image restoration". In: *Journal of Scientific Computing* 54.2-3 (2013), pp. 350–368.

[EB92] Jonathan Eckstein and Dimitri Bertsekas. "On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators". In: *Mathematical Programming* 55.1-3 (1992), pp. 293–318.

[Efr+04] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. "Least angle regression". In: *The Annals of statistics* 32.2 (2004), pp. 407–499.

[Elg+17] Ahmed Elgammal, Bingchen Liu, Mohamed Elhoseiny, and Marian Mazzone. "CAN: Creative Adversarial Networks, Generating" Art" by Learning About Styles and Deviating from Style Norms". In: *arXiv preprint arXiv:1706.07068* (2017).

[ES16] Ronen Eldan and Ohad Shamir. "The power of depth for feedforward neural networks". In: *COLT*. 2016, pp. 907–940.

[Ess+09] Ernie Esser, Xiaoqun Zhang, and Tony Chan. "A general framework for a class of first order primal-dual algorithms for TV minimization". In: *UCLA CAM Report* (2009), pp. 09–67.

[Ess09] Ernie Esser. "Applications of Lagrangian-based alternating direction methods and connections to split Bregman". In: *CAM report* 9 (2009), p. 31.

[Fan+15] Ethan X Fang, Bingsheng He, Han Liu, and Xiaoming Yuan. "Generalized alternating direction method of multipliers: new theoretical insights and applications". In: *Mathematical Programming Computation* 7.2 (2015), pp. 149–187.

[Fat08] Raanan Fattal. "Single image dehazing". In: *ACM TOG* 27.3 (2008), p. 72.

[Fat14] Raanan Fattal. "Dehazing using color-lines". In: *ACM TOG* 34.1 (2014), p. 13.

[FFD17] Li Fei-Fei and Jia Deng. "ImageNet: Where are we going? And where have we been?" In: *CVPR Workshop on Beyond ILSVRC* (2017).

[Fle05] Roger Fletcher. "On the Barzilai-Borwein method". In: *Optimization and control with applications*. Springer, 2005, pp. 235–256.

[Fri+16] Oriel Frigo, Neus Sabater, Julie Delon, and Pierre Hellier. "Split and match: Example-based adaptive patch sampling for unsupervised style transfer". In: *CVPR*. 2016, pp. 553–561.

[FSZ18] Shuicheng Yan Falong Shen and Gang Zeng. "Neural Style Transfer Via Meta Networks". In: *CVPR*. 2018.

[Gab83] Daniel Gabay. "Applications of the method of multipliers to variational inequalities". In: *Studies in mathematics and its applications* 15 (1983), pp. 299–331.

[Gat+15] Leon Gatys, Alexander S Ecker, and Matthias Bethge. "Texture synthesis using convolutional neural networks". In: *NIPS*. 2015.

[Gat+16] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks". In: *CVPR*. 2016.

[Gat+17] Leon A Gatys, Alexander S Ecker, Matthias Bethge, Aaron Hertzmann, and Eli Shechtman. "Controlling perceptual factors in neural style transfer". In: *CVPR*. 2017.

[GB16] Pontus Giselsson and Stephen Boyd. "Linear Convergence and Metric Selection in Douglas-Rachford Splitting and ADMM". In: (2016).

[Gha+15] Euhanna Ghadimi, André Teixeira, Iman Shames, and Mikael Johansson. "Optimal parameter selection for the alternating direction method of multipliers: quadratic problems". In: *IEEE Trans. Autom. Control* 60 (2015), pp. 644–658.

[Ghi+17] Golnaz Ghiasi, Honglak Lee, Manjunath Kudlur, Vincent Dumoulin, and Jonathon Shlens. "Exploring the structure of a real-time, arbitrary neural artistic stylization network". In: *BMVC* (2017).

[GM75] Roland Glowinski and A Marroco. "Sur l'approximation, par éléments finis d'ordre un, et la résolution, par pénalisation-dualité d'une classe de problémes de Dirichlet non linéaires". In: *ESAIM: Modlisation Mathmatique et Analyse Numrique* 9 (1975), pp. 41–76.

[GM76] Daniel Gabay and Bertrand Mercier. "A dual algorithm for the solution of nonlinear variational problems via finite element approximation". In: *Computers & Mathematics with Applications* 2.1 (1976), pp. 17–40.

[GO09] Tom Goldstein and Stanley Osher. "The split Bregman method for L1-regularized problems". In: *SIAM Journal on Imaging Sciences* 2.2 (2009), pp. 323–343.

[Gol+13] Donald Goldfarb, Shiqian Ma, and Katya Scheinberg. "Fast alternating linearization methods for minimizing the sum of two convex functions". In: *Mathematical Programming* 141.1-2 (2013), pp. 349–382.

[Gol+14a] Tom Goldstein, Christoph Studer, and Richard Baraniuk. "A Field Guide to Forward-Backward Splitting with a FASTA Implementation". In: *arXiv preprint arXiv:1411.3406* (2014).

[Gol+14b] Tom Goldstein, Brendan O'Donoghue, Simon Setzer, and Richard Baraniuk. "Fast alternating direction optimization methods". In: *SIAM Journal on Imaging Sciences* 7.3 (2014), pp. 1588–1623.

[Gol+15] Tom Goldstein, Min Li, and Xiaoming Yuan. "Adaptive primal-dual splitting methods for statistical learning and image processing". In: *NIPS*. 2015, pp. 2089–2097.

[Gol+16] Tom Goldstein, Gavin Taylor, Kawika Barabin, and Kent Sayre. "Unwrapping ADMM: Efficient Distributed Computing via Transpose Reduction". In: *AISTATS*. 2016.

[Goo+14a] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative adversarial nets". In: *NIPS*. 2014.

[Goo+14b] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. "Qualitatively characterizing neural network optimization problems". In: *ICLR* (2014).

[Goo+15] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *ICLR* (2015).

[Gu+18] Shuyang Gu, Congliang Chen, Jing Liao, and Lu Yuan. "Arbitrary Style Transfer with Deep Feature Reshuffle". In: *CVPR* (2018).

[HB17] Xun Huang and Serge Belongie. "Arbitrary Style Transfer in Real-Time With Adaptive Instance Normalization". In: *CVPR*. 2017, pp. 1501–1510.

[He+00] BS He, H Yang, and S Wang. "Alternating direction method with self-adaptive penalty parameters for monotone variational inequalities". In: *Jour. Optim. Theory and Appl.* 106.2 (2000), pp. 337–356.

[He+11] Kaiming He, Jian Sun, and Xiaoou Tang. "Single image haze removal using dark channel prior". In: *IEEE TPAMI* 33.12 (2011), pp. 2341–2353.

[He+15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *ICCV*. 2015, pp. 1026–1034.

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *CVPR*. 2016, pp. 770–778.

[Hin+15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[HL12] Mingyi Hong and Zhi-Quan Luo. "On the linear convergence of the alternating direction method of multipliers". In: *arXiv preprint arXiv:1208.3922* (2012).

[Hon+16] Mingyi Hong, Zhi-Quan Luo, and Meisam Razaviyayn. "Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems". In: *SIAM Journal on Optimization* 26.1 (2016), pp. 337–364.

[Hor+89] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural networks* 2.5 (1989), pp. 359–366.

[How+17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[Hsu+18] Yen-Chang Hsu, Zheng Xu, Zsolt Kira, and Jiawei Huang. "Learning to Cluster for Proposal-Free Instance Segmentation". In: *IJCNN*. IEEE. 2018, pp. 1–8.

[Hua+07] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Tech. rep. 07-49. University of Massachusetts, Amherst, 2007.

[Hua+15] Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. "Learning with a strong adversary". In: *arXiv preprint arXiv: 1511.03034* (2015).

[Hua+17] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. "Densely connected convolutional networks". In: *CVPR*. 2017.

[Hua+18] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. "Multimodal Unsupervised Image-to-Image Translation". In: *ECCV* (2018).

[HW17] Zehao Huang and Naiyan Wang. "Like What You Like: Knowledge Distill via Neuron Selectivity Transfer". In: *arXiv preprint arXiv:1707.01219* (2017).

[HY12a] Deren Han and Xiaoming Yuan. "A note on the alternating direction method of multipliers". In: *Journal of Optimization Theory and Applications* 155.1 (2012), pp. 227–238.

[HY12b] Bingsheng He and Xiaoming Yuan. "On the O(1/n) Convergence Rate of the Douglas-Rachford Alternating Direction Method". In: *SIAM Journal on Numerical Analysis* 50.2 (2012), pp. 700–709.

[HY15] Bingsheng He and Xiaoming Yuan. “On non-ergodic convergence rate of Douglas-Rachford alternating direction method of multipliers”. In: *Numerische Mathematik* 130 (2015), pp. 567–577.

[IS15] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *ICML*. 2015, pp. 448–456.

[Iso+17] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. “Image-to-image translation with conditional adversarial networks”. In: *CVPR* (2017).

[JG18] Daniel Jakubovitz and Raja Giryes. “Improving DNN robustness to adversarial attacks using Jacobian regularization”. In: *ECCV*. 2018, pp. 514–529.

[Jin+17] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. “Neural style transfer: A review”. In: *arXiv preprint arXiv:1705.04058* (2017).

[Jin+18] Yongcheng Jing, Yang Liu, Yezhou Yang, Zunlei Feng, Yizhou Yu, and Mingli Song. “Stroke Controllable Fast Style Transfer with Adaptive Receptive Fields”. In: *ECCV* (2018).

[Joh+16] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. “Perceptual losses for real-time style transfer and super-resolution”. In: *ECCV*. Springer. 2016, pp. 694–711.

[Kad+15] Mojtaba Kadkhodaie, Konstantina Christakopoulou, Maziar Sanjabi, and Arindam Banerjee. “Accelerated alternating direction method of multipliers”. In: *Proceedings of the 21th ACM SIGKDD*. 2015, pp. 497–506.

[Kan+18] Harini Kannan, Alexey Kurakin, and Ian Goodfellow. “Adversarial logit pairing”. In: *arXiv preprint arXiv:1803.06373* (2018).

[Kar+19] Tero Karras, Samuli Laine, and Timo Aila. “A style-based generator architecture for generative adversarial networks”. In: *CVPR* (2019).

[Kaw+17] Kenji Kawaguchi, Leslie Pack Kaelbling, and Yoshua Bengio. “Generalization in Deep Learning”. In: *arXiv preprint arXiv:1710.05468* (2017).

[Kaw16] Kenji Kawaguchi. “Deep Learning without Poor Local Minima”. In: *NIPS* (2016).

[KB14] Diederik Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *ICLR* (2014).

[Kes+16] 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”. In: *ICLR* (2016).

[KH09] Alex Krizhevsky and Geoffrey Hinton. “Learning multiple layers of features from tiny images”. In: (2009).

[Kim+17] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jungkwon Lee, and Jiwon Kim. "Learning to discover cross-domain relations with generative adversarial networks". In: *ICML* (2017).

[KK17] Seung Wook Kim and Hyo-Eun Kim. "Transferring Knowledge to Smaller Network with Class-Distance Loss". In: *ICLR Workshop* (2017).

[Kov+15] Artiom Kovnatsky, Klaus Glashoff, and Michael M Bronstein. "MADMM: a generic algorithm for non-smooth optimization on manifolds". In: *arXiv preprint arXiv:1505.07676* (2015).

[Kra+13] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. "3D Object Representations for Fine-Grained Categorization". In: *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia, 2013.

[Kri+12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *NIPS*. 2012, pp. 1097–1105.

[Kur+16a] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *arXiv preprint arXiv:1607.02533* (2016).

[Kur+16b] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial machine learning at scale". In: *ICLR* (2016).

[KW13] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[Lam+18] Alex Lamb, Jonathan Binas, Anirudh Goyal, Dmitriy Serdyuk, Sandeep Subramanian, Ioannis Mitliagkas, and Yoshua Bengio. "Fortified networks: Improving the robustness of deep networks by modeling the manifold of hidden representations". In: *arXiv preprint arXiv:1804.02485* (2018).

[LeC+98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[Lee+06] Su-In Lee, Honglak Lee, Pieter Abbeel, and Andrew Ng. "Efficient L1 regularized logistic regression". In: *AAAI*. Vol. 21. 2006, p. 401.

[Li+17a] Boyi Li, Xiulian Peng, Zhangyang Wang, Jizheng Xu, and Dan Feng. "Aod-net: All-in-one dehazing network". In: *ICCV*. 2017, pp. 4770–4778.

[Li+17b] Boyi Li, Wenqi Ren, Dengpan Fu, Dacheng Tao, Dan Feng, Wenjun Zeng, and Zhangyang Wang. "RESIDE: A Benchmark for Single Image Dehazing". In: *arXiv preprint arXiv:1712.04143* (2017).

[Li+17c] Hao Li, Soham De, Zheng Xu, Christoph Studer, Hanan Samet, and Tom Goldstein. "Training Quantized Nets: A Deeper Understanding". In: *NIPS* (2017).

[Li+17d] Wen Li, Zheng Xu, Dong Xu, Dengxin Dai, and Luc Van Gool. "Domain Generalization and Adaptation using Low Rank Exemplar SVMs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2017).

[Li+17e] Yanghao Li, Naiyan Wang, Jiaying Liu, and Xiaodi Hou. "Demystifying neural style transfer". In: *IJCAI* (2017).

[Li+17f] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. "Universal style transfer via feature transforms". In: *NIPS*. 2017, pp. 385–395.

[Li+18a] Chongyi Li, Jichang Guo, Fatih Porikli, Huazhu Fu, and Yanwei Pang. "A Cascaded Convolutional Neural Network for Single Image Dehazing". In: *arXiv preprint arXiv:1803.07955* (2018).

[Li+18b] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. "Visualizing the Loss Landscape of Neural Nets". In: *NeurIPS* (2018).

[Li+18c] Yijun Li, Ming-Yu Liu, Xueting Li, Ming-Hsuan Yang, and Jan Kautz. "A Closed-form Solution to Photorealistic Image Stylization". In: *ECCV* (2018).

[Lia+17] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. "Visual attribute transfer through deep image analogy". In: *ACM (TOG)* 36.4 (2017), p. 120.

[Lin+11] Zhouchen Lin, Risheng Liu, and Zhixun Su. "Linearized alternating direction method with adaptive penalty for low-rank representation". In: *NIPS*. 2011, pp. 612–620.

[Lin+15] Tianyi Lin, Shiqian Ma, and Shuzhong Zhang. "On the global linear convergence of the ADMM with multiblock variables". In: *SIAM Journal on Optimization* 25.3 (2015), pp. 1478–1497.

[Liu+09] Jun Liu, Jianhui Chen, and Jieping Ye. "Large-scale sparse logistic regression". In: *ACM SIGKDD*. 2009, pp. 547–556.

[Liu+13] Risheng Liu, Zhouchen Lin, and Zhixun Su. "Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning." In: *ACML*. 2013, pp. 116–132.

[Liu+15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Attributes in the Wild". In: *ICCV*. 2015.

[Liu+17] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. "Unsupervised Image-to-Image Translation Networks". In: *NIPS* (2017).

[LN89] Dong C Liu and Jorge Nocedal. “On the limited memory BFGS method for large scale optimization”. In: *Mathematical programming* 45.1 (1989), pp. 503–528.

[LO14] Rongjie Lai and Stanley Osher. “A splitting method for orthogonality constrained problems”. In: *Journal of Scientific Computing* 58.2 (2014), pp. 431–449.

[LP15] Guoyin Li and Ting Kei Pong. “Global convergence of splitting methods for nonconvex composite optimization”. In: *SIAM Journal on Optimization* 25.4 (2015), pp. 2434–2460.

[LS15] Athanasios P Liavas and Nicholas D Sidiropoulos. “Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers”. In: *IEEE Transactions on Signal Processing* 63.20 (2015), pp. 5450–5463.

[LS17] Shiyu Liang and R Srikant. “Why Deep Neural Networks for Function Approximation?” In: *ICLR* (2017).

[Lu+17] Jiajun Lu, Hussein Sibai, Evan Fabry, and David Forsyth. “No need to worry about adversarial examples in object detection in autonomous vehicles”. In: *arXiv preprint arXiv:1707.03501* (2017).

[Luo+16] Ping Luo, Zhenyao Zhu, Ziwei Liu, Xiaogang Wang, Xiaoou Tang, et al. “Face Model Compression by Distilling Knowledge from Neurons.” In: *AAAI*. 2016, pp. 3560–3566.

[LW16a] Chuan Li and Michael Wand. “Combining markov random fields and convolutional neural networks for image synthesis”. In: *CVPR*. 2016, pp. 2479–2486.

[LW16b] Chuan Li and Michael Wand. “Precomputed real-time texture synthesis with markovian generative adversarial networks”. In: *ECCV*. Springer. 2016, pp. 702–716.

[LZ15] Guanghui Lan and Yi Zhou. “An optimal randomized incremental gradient method”. In: *arXiv preprint arXiv:1507.02000* (2015).

[Ma+18] Xingjun Ma, Bo Li, Yisen Wang, Sarah M Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E Houle, and James Bailey. “Characterizing adversarial subspaces using local intrinsic dimensionality”. In: *arXiv preprint arXiv:1801.02613* (2018).

[Mad+17] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. “Towards deep learning models resistant to adversarial attacks”. In: *ICLR* (2017).

[Mao+16] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. “Least squares generative adversarial networks”. In: *arXiv preprint* (2016).

[MC17] Dongyu Meng and Hao Chen. "Magnet: a two-pronged defense against adversarial examples". In: *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2017, pp. 135–147.

[McC76] Earl J McCartney. "Optics of the atmosphere: scattering by molecules and particles". In: *New York, John Wiley and Sons, Inc., 1976. 421 p.* (1976).

[MD+16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks". In: *CVPR*. 2016, pp. 2574–2582.

[MD+17a] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. "Analysis of universal adversarial perturbations". In: *arXiv preprint arXiv:1705.09554* (2017).

[MD+17b] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. "Universal Adversarial Perturbations". In: *CVPR*. 2017, pp. 1765–1773.

[Men+13] Gaofeng Meng, Ying Wang, Jiangyong Duan, Shiming Xiang, and Chunhong Pan. "Efficient image dehazing with boundary constraint and contextual regularization". In: *ICCV*. IEEE. 2013, pp. 617–624.

[Met+16] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. "Unrolled generative adversarial networks". In: *arXiv preprint arXiv:1611.02163* (2016).

[Met+17] Jan Hendrik Metzen, Mummadi Chaithanya Kumar, Thomas Brox, and Volker Fischer. "Universal adversarial perturbations against semantic image segmentation". In: *ICCV*. 2017.

[Mik+14] Ondrej Miksik, Vibhav Vineet, Patrick Pérez, Philip HS Torr, and FR Cesson Sévigné. "Distributed non-convex admm-inference in large-scale random fields". In: *British Machine Vision Conference, BMVC*. 2014.

[MK18] Takeru Miyato and Masanori Koyama. "cGANs with projection discriminator". In: *ICLR* (2018).

[Mop+17] Konda Reddy Mopuri, Utsav Garg, and R Venkatesh Babu. "Fast feature fool: A data independent approach to universal adversarial perturbations". In: *BMVC* (2017).

[Nem+09] Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. "Robust stochastic approximation approach to stochastic programming". In: *SIAM Journal on optimization* 19.4 (2009), pp. 1574–1609.

[Ngu+15] Anh Nguyen, Jason Yosinski, and Jeff Clune. "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images". In: *CVPR*. 2015, pp. 427–436.

[Nis+15] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. Jordan. "A General Analysis of the Convergence of ADMM". In: *ICML*. 2015.

[NN02] Srinivasa G Narasimhan and Shree K Nayar. "Vision and the atmosphere". In: *International Journal of Computer Vision* 48.3 (2002), pp. 233–254.

[Ode+17] Augustus Odena, Christopher Olah, and Jonathon Shlens. "Conditional image synthesis with auxiliary classifier gans". In: *ICML* (2017).

[Pap+16] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. "The limitations of deep learning in adversarial settings". In: *EuroS&P*. IEEE. 2016, pp. 372–387.

[PB16] Balamurugan Palaniappan and Francis Bach. "Stochastic Variance Reduction Methods for Saddle-Point Problems". In: *NIPS*. 2016, pp. 1408–1416.

[Per+18] Julien Perolat, Mateusz Malinowski, Bilal Piot, and Olivier Pietquin. "Playing the Game of Universal Adversarial Perturbations". In: *arXiv preprint arXiv:1809.07802* (2018).

[Phi+07] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. "Object Retrieval with Large Vocabularies and Fast Spatial Matching". In: *CVPR*. 2007.

[Pou+18] Omid Poursaeed, Isay Katsman, Bicheng Gao, and Serge Belongie. "Generative Adversarial Perturbations". In: *CVPR* (2018).

[Qia+16] Linbo Qiao, Tianyi Lin, Yu-Gang Jiang, Fan Yang, Wei Liu, and Xicheng Lu. "On Stochastic Primal-Dual Hybrid Gradient Approach for Compositely Regularized Minimization". In: *ECAI*. 2016.

[Rad+16] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *ICLR* (2016).

[Rag+18] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. "Certified defenses against adversarial examples". In: *arXiv preprint arXiv:1801.09344* (2018).

[Ras+16] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. "XNOR-net: Imagenet classification using binary convolutional neural networks". In: *ECCV*. Springer. 2016, pp. 525–542.

[RDC14] Arvind Raghunathan and Stefano Di Cairano. "Alternating direction method of multipliers for strictly convex quadratic programs: Optimal parameter selection". In: *American Control Conf.* 2014, pp. 4324–4329.

[Ren+16] Wenqi Ren, Si Liu, Hua Zhang, Jinshan Pan, Xiaochun Cao, and Ming-Hsuan Yang. "Single image dehazing via multi-scale convolutional neural networks". In: *ECCV*. Springer. 2016, pp. 154–169.

[Ren+18] Wenqi Ren, Lin Ma, Jiawei Zhang, Jinshan Pan, Xiaochun Cao, Wei Liu, and Ming-Hsuan Yang. "Gated fusion network for single image dehazing". In: *CVPR* (2018).

[RM+18a] Konda Reddy Mopuri, Phani Krishna Uppala, and R Venkatesh Babu. "Ask, Acquire, and Attack: Data-free UAP Generation using Class Impressions". In: *ECCV*. 2018, pp. 19–34.

[RM+18b] Konda Reddy Mopuri, Utkarsh Ojha, Utsav Garg, and R Venkatesh Babu. "NAG: Network for adversary generation". In: *CVPR*. 2018, pp. 742–751.

[Roc70] R. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[Rom+15] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. "Fitnets: Hints for thin deep nets". In: *ICLR* (2015).

[Ron+15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *MICCAI*. Springer. 2015, pp. 234–241.

[Roy+17] Amélie Royer, Konstantinos Bousmalis, Stephan Gouws, Fred Bertsch, Inbar Moressi, Forrester Cole, and Kevin Murphy. "XGAN: Unsupervised Image-to-Image Translation for many-to-many Mappings". In: *arXiv preprint arXiv:1711.05139* (2017).

[Rud+92] Leonid I Rudin, Stanley Osher, and Emad Fatemi. "Nonlinear total variation based noise removal algorithms". In: *Physica D: Nonlinear Phenomena* 60.1 (1992), pp. 259–268.

[Rus+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. "Imagenet large scale visual recognition challenge". In: *IJCV* 115.3 (2015), pp. 211–252.

[Sak+17] Christos Sakaridis, Dengxin Dai, and Luc Van Gool. "Semantic Foggy Scene Understanding with Synthetic Data". In: *arXiv preprint arXiv:1708.07819* (2017).

[Sam+19] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. "Defense-gan: Protecting classifiers against adversarial attacks using generative models". In: *ICLR* (2019).

[San+19] Karthik A Sankararaman, Soham De, Zheng Xu, W Ronny Huang, and Tom Goldstein. "The Impact of Neural Network Overparameterization on Gradient Confusion and Stochastic Gradient Descent". In: *arXiv preprint arXiv:1904.06963* (2019).

[Sch+07] Mark Schmidt, Glenn Fung, and Rómer Rosales. "Fast optimization methods for l1 regularization: A comparative study and two new approaches". In: *ECML*. Springer, 2007, pp. 286–297.

[Sch+14] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nešić, Xi Wang, and Porter Westling. "High-resolution stereo datasets with subpixel-accurate ground truth". In: *German Conference on Pattern Recognition*. Springer. 2014, pp. 31–42.

[Sch+18] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. "Adversarially robust generalization requires more data". In: *NeurIPS*. 2018, pp. 5014–5026.

[SF14] Dennis L Sun and Cédric Févotte. "Alternating direction method of multipliers for non-negative matrix factorization with the beta-divergence". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pp. 6201–6205.

[Sha+15] Uri Shaham, Yutaro Yamada, and Sahand Negahban. "Understanding adversarial training: Increasing local stability of neural nets through robust optimization". In: *arXiv preprint arXiv:1511.05432* (2015).

[Sha+18] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S Davis, and Tom Goldstein. "Universal Adversarial Training". In: *arXiv preprint arXiv:1811.11304* (2018).

[Sha+19] Ali Shafahi, Mahyar Najibi, Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. "Adversarial Training for Free". In: *arXiv preprint* (2019).

[She+16] Jonathan Shen, Noranart Vesdapunt, Vishnu N Boddeti, and Kris M Kitani. "In Teacher We Trust: Learning Compressed Models for Pedestrian Detection". In: *arXiv preprint arXiv:1612.00478* (2016).

[She+18] Lu Sheng, Ziyi Lin, Jing Shao, and Xiaogang Wang. "Avatar-Net: Multi-scale Zero-shot Style Transfer by Feature Decoration". In: *CVPR* (2018).

[Sil+12] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. "Indoor segmentation and support inference from rgbd images". In: *ECCV* (2012), pp. 746–760.

[Sin+18] Aman Sinha, Hongseok Namkoong, and John Duchi. "Certifying some distributional robustness with principled adversarial training". In: *ICLR* (2018).

[Son+16] Changkyu Song, Sejong Yoon, and Vladimir Pavlovic. "Fast ADMM Algorithm for Distributed Optimization with Adaptive Penalty". In: *AAAI* (2016).

[Sri+14] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting." In: *JMLR* 15.1 (2014), pp. 1929–1958.

[SS17] Itay Safran and Ohad Shamir. "Depth-Width Tradeoffs in Approximating Natural Functions with Neural Networks". In: *ICML*. 2017, pp. 2979–2987.

[ST17] Atsushi Shibagaki and Ichiro Takeuchi. "Stochastic Primal Dual Coordinate Method with Non-Uniform Sampling Based on Optimality Violations". In: *arXiv preprint arXiv:1703.07056* (2017).

[Su+18] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. "Is Robustness the Cost of Accuracy?–A Comprehensive Study on the Robustness of 18 Deep Image Classification Models". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 631–648.

[SZ14] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[Sze+13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks". In: *ICLR* (2013).

[Sze+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. "Rethinking the inception architecture for computer vision". In: *CVPR*. 2016, pp. 2818–2826.

[Tai+17] Yaniv Taigman, Adam Polyak, and Lior Wolf. "Unsupervised cross-domain image generation". In: *ICLR* (2017).

[Tan+14] Ketan Tang, Jianchao Yang, and Jue Wang. "Investigating haze-relevant features in a learning framework for image dehazing". In: *CVPR*. 2014, pp. 2995–3000.

[Tan08] Robby T Tan. "Visibility in bad weather from a single image". In: *CVPR*. 2008.

[Tay+16] Gavin Taylor, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein. "Training Neural Networks Without Gradients: A Scalable ADMM Approach". In: *ICML* (2016).

[Tay+17] Gavin Taylor, Zheng Xu, and Tom Goldstein. "Scalable Classifiers With ADMM and Transpose Reduction". In: *AAAI workshop on distributed machine learning*. 2017.

[Teh+17] Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. "Distral: Robust Multitask Reinforcement Learning". In: *arXiv preprint arXiv:1707.04175* (2017).

[Tel16] Matus Telgarsky. "Benefits of depth in neural networks". In: *arXiv preprint arXiv:1602.04485* (2016).

[TH09] Jean-Philippe Tarel and Nicolas Hautiere. "Fast visibility restoration from a single color or gray level image". In: *ICCV*. IEEE. 2009, pp. 2201–2208.

[Tje+17] Vincent Tjeng, Kai Xiao, and Russ Tedrake. "Evaluating Robustness of Neural Networks with Mixed Integer Programming". In: *arXiv preprint arXiv:1711.07356* (2017).

[Tra+17] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. "Ensemble adversarial training: Attacks and defenses". In: *arXiv preprint arXiv:1705.07204* (2017).

[Tsi+18] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. "Robustness may be at odds with accuracy". In: *ICLR* 1050 (2018), p. 11.

[TY11] Min Tao and Xiaoming Yuan. "Recovering low-rank and sparse components of matrices from incomplete and noisy observations". In: *SIAM Journal on Optimization* 21.1 (2011), pp. 57–81.

[TY16a] Min Tao and Xiaoming Yuan. "Convergence analysis of the direct extension of ADMM for multiple-block separable convex minimization". In: *arXiv preprint arXiv:1609.07221* (2016).

[TY16b] Wenyi Tian and Xiaoming Yuan. "Faster Alternating Direction Method of Multipliers with a Worst-case O $(1/n^2)$ Convergence Rate". In: (2016).

[TZ15] Da Tang and Tong Zhang. "On the Duality Gap Convergence of ADMM Methods". In: *arXiv preprint arXiv:1508.03702* (2015).

[Uly+16a] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. "Instance Normalization: The Missing Ingredient for Fast Stylization". In: *CoRR* abs/1607.08022 (2016).

[Uly+16b] Dmitry Ulyanov, Vadim Lebedev, Andrea Vedaldi, and Victor S Lempitsky. "Texture Networks: Feed-forward Synthesis of Textures and Stylized Images." In: *ICML*. 2016, pp. 1349–1357.

[Uly+17a] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. "Deep Image Prior". In: *CoRR* abs/1711.10925 (2017).

[Uly+17b] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. "Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis". In: *CVPR*. 2017.

[Urb+17] Gregor Urban, Krzysztof J Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson. "Do Deep Convolutional Nets Really Need to be Deep and Convolutional?" In: *ICLR* (2017).

[Wan+14] Fenghui Wang, Zongben Xu, and Hong-Kun Xu. "Convergence of Bregman alternating direction method with multipliers for nonconvex composite problems". In: *arXiv preprint arXiv:1410.8625* (2014).

[Wan+15] Yu Wang, Wotao Yin, and Jinshan Zeng. "Global convergence of ADMM in nonconvex nonsmooth optimization". In: *arXiv preprint arXiv:1511.06324* (2015).

[Wan+16] Jingdong Wang, Zhen Wei, Ting Zhang, and Wenjun Zeng. "Deeply-fused nets". In: *arXiv preprint arXiv:1605.07716* (2016).

[Wan+17] Xin Wang, Geoffrey Oxholm, Da Zhang, and Yuan-Fang Wang. "Multimodal Transfer: A Hierarchical Deep Convolutional Neural Network for Fast Artistic Style Transfer". In: *CVPR*. 2017, pp. 5239–5247.

[Wan+18] Shiqi Wang, Yizheng Chen, Ahmed Abdou, and Suman Jana. "Mixtrain: Scalable training of formally robust neural networks". In: *arXiv preprint arXiv:1811.02625* (2018).

[WC16] Mengdi Wang and Yichen Chen. "An online primal-dual method for discounted Markov decision processes". In: *Decision and Control (CDC), 2016 IEEE 55th Conference on*. IEEE. 2016, pp. 4516–4521.

[Wel+10] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. *Caltech-UCSD Birds 200*. Tech. rep. CNS-TR-2010-001. California Institute of Technology, 2010.

[Wen+10] Zaiwen Wen, Donald Goldfarb, and Wotao Yin. "Alternating direction augmented Lagrangian methods for semidefinite programming". In: *Mathematical Programming Computation* 2.3-4 (2010), pp. 203–230.

[Wen+12] Zaiwen Wen, Chao Yang, Xin Liu, and Stefano Marchesini. "Alternating direction methods for classical and ptychographic phase retrieval". In: *Inverse Problems* 28.11 (2012), p. 115010.

[Wil+17a] Michael J. Wilber, Chen Fang, Hailin Jin, Aaron Hertzmann, John Collomosse, and Serge Belongie. "BAM! The Behance Artistic Media Dataset for Recognition Beyond Photography". In: *ICCV*. 2017.

[Wil+17b] Pierre Wilmot, Eric Risser, and Connelly Barnes. "Stable and controllable neural texture synthesis and style transfer using histogram losses". In: *arXiv preprint arXiv:1701.08893* (2017).

[Won+18] Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. "Scaling provable adversarial defenses". In: *NeurIPS*. 2018, pp. 8400–8409.

[Wri+09a] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. "Robust face recognition via sparse representation". In: *IEEE Trans. Pattern Analysis and Machine Intelligence* 31 (2009), pp. 210–227.

[Wri+09b] John Wright, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma. "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization". In: *Advances in neural information processing systems*. 2009, pp. 2080–2088.

[Wri+09c] Stephen Wright, Robert Nowak, and Mário Figueiredo. "Sparse reconstruction by separable approximation". In: *IEEE Trans. Signal Processing* 57 (2009), pp. 2479–2493.

[WX17] Jialei Wang and Lin Xiao. "Exploiting Strong Convexity from Data with Primal-Dual First-Order Algorithms". In: *ICML* (2017).

[Xia+18] Chaowei Xiao, Bo Li, Jun-Yan Zhu, Warren He, Mingyan Liu, and Dawn Song. "Generating adversarial examples with adversarial networks". In: *IJCAI* (2018).

[Xia+19] Kai Xiao, Vincent Tjeng, Nur Muhammad Shafiullah, and Aleksander Madry. "Training for Faster Adversarial Robustness Verification via Inducing ReLU Stability". In: *ICLR* (2019).

[Xie+19] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan Yuille, and Kaiming He. "Feature denoising for improving adversarial robustness". In: *CVPR* (2019).

[Xu+12] Yangyang Xu, Wotao Yin, Zaiwen Wen, and Yin Zhang. "An alternating direction algorithm for matrix completion with nonnegative factors". In: *Frontiers of Mathematics in China* 7.2 (2012), pp. 365–384.

[Xu+14] Zheng Xu, Wen Li, Li Niu, and Dong Xu. "Exploiting Low-Rank Structure from Latent Domains for Domain Generalization". In: *ECCV*. 2014.

[Xu+15] Zheng Xu, Xue Li, Kuiyuan Yang, and Tom Goldstein. "Exploiting Low-rank Structure for Discriminative Sub-categorization". In: *BMVC, Swansea, UK, September 7-10, 2015*. 2015.

[Xu+16a] Zheng Xu, Soham De, Mário A. T. Figueiredo, Christoph Studer, and Tom Goldstein. "An Empirical Study of ADMM for Nonconvex Problems". In: *NIPS workshop on nonconvex optimization*. 2016.

[Xu+16b] Zheng Xu, Furong Huang, Louiqa Raschid, and Tom Goldstein. "Nonnegative Factorization of the Occurrence Tensor from Financial Contracts". In: *NIPS workshop on tensor methods*. 2016.

[Xu+17a] Weilin Xu, David Evans, and Yanjun Qi. "Feature squeezing: Detecting adversarial examples in deep neural networks". In: *arXiv preprint arXiv:1704.01155* (2017).

[Xu+17b] Zheng Xu, Mario AT Figueiredo, and Tom Goldstein. "Adaptive ADMM with Spectral Penalty Parameter Selection". In: *AISTATS* (2017).

[Xu+17c] Zheng Xu, Gavin Taylor, Hao Li, Mario AT Figueiredo, Xiaoming Yuan, and Tom Goldstein. "Adaptive Consensus ADMM for Distributed Optimization". In: *ICML* (2017).

[Xu+17d] Zheng Xu, Mario AT Figueiredo, Xiaoming Yuan, Christoph Studer, and Tom Goldstein. "Adaptive Relaxed ADMM: Convergence Theory and Practical Implementation". In: *CVPR* (2017).

[Xu+17e] Zheng Xu, Mario AT Figueiredo, and Tom Goldstein. "Practical Guide to Penalty Parameter Selection for ADMM". In: *journal draft in preparation* (2017).

[Xu+18a] Zheng Xu, Xitong Yang, Xue Li, and Xiaoshuai Sun. "Strong baseline for single image dehazing with deep features and instance normalization". In: *BMVC*. Vol. 2. 3. 2018, p. 5.

[Xu+18b] Zheng Xu, Yen-Chang Hsu, and Jiawei Huang. "Training Student Networks for Acceleration with Conditional Adversarial Networks". In: *BMVC* (2018).

[Xu+19a] Zheng Xu, Michael Wilber, Chen Fang, Aaron Hertzmann, and Hailin Jin. "Beyond textures: Learning from multi-domain artistic images for arbitrary style transfer". In: *Expressive* (2019).

[Xu+19b] Zheng Xu, Ali Shafahi, and Tom Goldstein. "Exploiting Adaptive Network for Robustness". In: *anonymous conference submission* (2019).

[Yad+18] Abhay Yadav, Sohil Shah, Zheng Xu, David Jacobs, and Tom Goldstein. "Stabilizing Adversarial Nets With Prediction Methods". In: *ICLR* (2018).

[Yan+11] Chao Yang, Jianliang Qian, Andre Schirotzek, Filipe Maia, and Stefano Marchesini. "Iterative algorithms for ptychographic phase retrieval". In: *arXiv preprint arXiv:1105.5628* (2011).

[Yan+18] Xitong Yang, Zheng Xu, and Jiebo Luo. "Towards Perceptual Image Dehazing by Physics-based Disentanglement and Adversarial Training". In: *AAAI* (2018).

[Yi+17] Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. "DualGAN: Unsupervised Dual Learning for Image-To-Image Translation". In: *CVPR*. 2017, pp. 2849–2857.

[Yim+17] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. "A Gift from Knowledge Distillation: Fast Optimization, Network Minimization and Transfer Learning". In: *CVPR* (2017).

[You+17] Shan You, Chang Xu, Chao Xu, and Dacheng Tao. "Learning from Multiple Teacher Networks". In: *KDD*. ACM. 2017, pp. 1285–1294.

[Yu+15] Adams Wei Yu, Qihang Lin, and Tianbao Yang. "Doubly Stochastic Primal-Dual Coordinate Method for Empirical Risk Minimization and Bilinear Saddle-Point Problem". In: *arXiv preprint arXiv:1508.03390* (2015).

[YY13] Junfeng Yang and Xiaoming Yuan. "Linearized augmented Lagrangian and alternating direction methods for nuclear norm minimization". In: *Mathematics of Computation* 82.281 (2013), pp. 301–329.

[ZC08] Mingqiang Zhu and Tony Chan. "An efficient primal-dual hybrid gradient algorithm for total variation image restoration". In: *UCLA CAM Report* (2008), pp. 08–34.

[ZD17] Hang Zhang and Kristin Dana. "Multi-style generative network for real-time transfer". In: *arXiv preprint arXiv:1703.06953* (2017).

[ZH05] Hui Zou and Trevor Hastie. "Regularization and variable selection via the elastic net". In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320.

[Zha+17a] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. "Understanding deep learning requires rethinking generalization". In: *ICLR* (2017).

[Zha+17b] He Zhang, Vishwanath Sindagi, and Vishal M Patel. "Joint transmission map estimation and dehazing using deep networks". In: *arXiv preprint arXiv:1708.00581* (2017).

[Zha+18a] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *CVPR* (2018).

[Zha+18b] Yexun Zhang, Wenbin Cai, and Ya Zhang. "Separating Style and Content for Generalized Style Transfer". In: *CVPR* (2018).

[Zha+19a] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P Xing, Laurent El Ghaoui, and Michael I Jordan. "Theoretically Principled Trade-off between Robustness and Accuracy". In: *ICML* (2019).

[Zha+19b] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. "Fixup Initialization: Residual Learning Without Normalization". In: *ICLR* (2019).

[Zho+06] Bin Zhou, Li Gao, and Yu-Hong Dai. "Gradient methods with adaptive step-sizes". In: *Computational Optimization and Applications* 35 (2006), pp. 69–86.

[Zho+10] Zihan Zhou, Xiaodong Li, John Wright, Emmanuel Candes, and Yi Ma. "Stable principal component pursuit". In: *2010 IEEE International Symposium on Information Theory*. IEEE. 2010, pp. 1518–1522.

[Zhu+15] Qingsong Zhu, Jiaming Mai, and Ling Shao. "A fast single image haze removal algorithm using color attenuation prior". In: *IEEE TIP* 24.11 (2015), pp. 3522–3533.

[Zhu+17a] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. "Toward multimodal image-to-image translation". In: *NIPS*. 2017, pp. 465–476.

[Zhu+17b] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. "Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks". In: *CVPR*. 2017, pp. 2223–2232.

[ZK14] Ruiliang Zhang and James T Kwok. "Asynchronous Distributed ADMM for Consensus Optimization." In: *ICML*. 2014, pp. 1701–1709.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. "Wide residual networks". In: *arXiv preprint arXiv:1605.07146* (2016).

[ZK17] Sergey Zagoruyko and Nikos Komodakis. "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer". In: *ICLR* (2017).

[ZL15] Yuchen Zhang and Xiao Lin. "Stochastic Primal-Dual Coordinate Method for Regularized Empirical Risk Minimization." In: *ICML*. 2015, pp. 353–361.

[ZP18] He Zhang and Vishal M Patel. "Densely Connected Pyramid Dehazing Network". In: *CVPR* (2018).

[ZS15] Zhanxing Zhu and Amos J Storkey. "Adaptive stochastic primal-dual coordinate descent for separable saddle point problems". In: *ECML-PKDD*. 2015, pp. 645–658.

[ZS16] Zhanxing Zhu and Amos J Storkey. "Stochastic parallel block coordinate descent for large-scale saddle point problems". In: *AAAI*. 2016.