Self-Similar Shock Formation and Some Results in Deep Learning

By

KYLE ROBERT CHICKERING

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Applied Mathematics

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Muhao Chen, Chair

F. Javier Arsuaga

Yunpeng Shi

Committee in Charge

2025

To my wife, Shannon.

—

For Nicholas Juárez (1996 - 2012), promise unrealized.

—

For Ernie Vollmer (1937 - 2025), who couldn’t see me finish.

Contents

Abstract v

Acknowledgments vii

Chapter 1. Introduction 1

1.1. A New Perspective on Overfitting MLP Networks 2

1.2. Some Applications of Overfitting MLPs 5

1.3. Outline 7

Chapter 2. Asymptotically Self-Similar Shock Formation for the Fractal Burgers Equation 8

2.1. Introduction 8

2.2. Self-Similar Transformation, Initial Data, and Statement of Theorem 15

2.3. Bootstraps: Assumptions and Consequences 20

2.4. Bootstraps: Closing The L^∞ Estimates 27

2.5. Proof of Theorem 36

Chapter 3. Computational Efficiency: n-Tangent Prop 47

3.1. Introduction 47

3.2. Physics Informed Neural Networks 49

3.3. n-TP 50

3.4. Empirical Results 56

3.5. Conclusions 67

Chapter 4. Statistical Efficiency: QLIP & Enhanced Vision Priors 69

4.1. Introduction 69

4.2. Why CLIP Fails at Higher Resolutions 71

4.3. Addressing the Mesoscopic and Interpolation Biases 73

4.4. Experimental Results 80

4.5. Ablations 92

4.6. Related Work 94
4.7. Conclusion 95
Limitations 95

Bibliography 97

Abstract

Modern mathematical practice is often informed by computational methods, and conversely, improvements in computational methods can be informed by mathematics. In this thesis we pull on these rich threads and present an interdisciplinary research program which begins with the mathematical analysis of shock formation for the fractal Burgers equation and culminates in applying intuitions gleaned from our research program on fluid mechanics to improve the statistical efficiency of multi-modal large language models.

We begin with a problem from pure analysis in Chapter 2: constructing and precisely describing the shock formation process for the fractal Burgers equation. Solving this problem requires extending existing theory to cover fractional dissipation. Our result is the first to successfully deal with the fractional dissipation term through a careful pointwise analysis of the fractional Laplacian.

During the completion of this first result, we made frequent use of numerical simulation to inform our analytic methodology. We show that the dynamics of shock formation converge asymptotically to the stable self-similar Burgers dynamics. A thorough analysis of these dynamics is challenging using traditional numerical methods, however recently authors have shown that physics-informed neural networks (PINNS) are well-suited to solving these often unstable numerical problems. In our studies using PINNs we quickly realized that the standard technique of applying autodifferentiation introduces unnecessary repeated computation. In Chapter 3 we connect the PINN training scenario to a classical machine learning algorithm through a novel insight. We show that a simple extension of the TangentProp algorithm can reduce the runtime of PINN training from exponential to quasilinar and we run empirical experiments to demonstrate the efficacy of our theory.

From our work on PINNs we developed a novel viewpoint regarding what we call overfitting MLP networks. We demonstrate the payoff of this perspective by applying overfitting MLPs to a problem from the large-language modeling literature. After our work on PINNs, we began to explore ways in which we could apply our ideas to improving the performance of large language models. We observed that multi-modal vision language models use inefficient priors to ingest images, and that this ingestion process can be streamlined using the theory and intuitions developed during our study of PINNs. In particular, the overfitting perspective on PINNs frames the learning target as reducing training loss, subject to some auxiliary condition. In the realm MLLMs we show that the standard method for encoding images can be improved by using a content aware selection strategy. However using this selection strategy introduces an interpolation problem which must be solved. This problem cannot be solved satisfactorily using standard bilinear or bicubic interpolation, but it can be phrased as an MLP overfitting problem with an auxiliary condition, exactly

like a PINN. In Chapter 4 we are able to leverage our intuitions from studying PINNs and apply them in this use-case as well to train overfit interpolation networks and dramatically improve MLLM inference on a challenging benchmark without having to retrain or fine-tune the underlying LLM. Our solution to this problem demonstrates the advantage of leveraging mathematical understanding to improve the statistical efficiency of a model, rather than relying purely on compute (re-training).

Acknowledgments

Thank you to my wife, Shannon, for supporting me through long nights, uninspired lows, and always reminding me that I could finish.

Thank you to my family: Mom, Dad, Ryan, Ona, Roger & Alison, Nanna & Papa.

To my friends and fellow graduate students who kept me on track, I am thankful for your companionship and support. In no particular order: Gavin Pandya, Ryan-Chris Moreno-Vasquez, Michael Ragone, Jeff Nichols, Noah Wiesner, Ken Brown, Eli & Gal Moore, Christian & Oliva Wingate, Daniel Stuziak, Nick Paull, and Adam Wentland.

Finally, thank you to my mentors and advisors who helped me navigate this crazy journey: Muhao Chen, Javier Arsuaga, Tim Lewis, Joseph Biello, Randee & Terry Reidy, Leslee Fournier, Patrice Simard, David Raskino, Heather Russell, Aleksey Golovinskiy, Minh Hoang, and Tina Denena.

CHAPTER 1

Introduction

This thesis follows a research trajectory which initially starts in the study of partial differential equations and ends up having practical applications in multi-modal large language modeling. At first glance these fields may seem unrelated, but we will demonstrate that a numerical research program which grew out of our analytic work in partial differential equations ends up yielding tangible applications in improving the statistical efficiency of large language modeling. There are two themes which unify this work. The first theme is the interplay between pure mathematical practice and improvements of numerical or applied mathematics. We use numerical methods to inform our work in pure mathematics, and, as this thesis demonstrates, our work in pure mathematics can lead to research advancements in unexpected fields. The second thematic flavor of this thesis is doing more with the same amount of resources. In Chapter 2 we prove highly detailed analytic bounds on the shock solutions to fractal Burgers equation without having to prove new and restrictive bounds on the solutions. In Chapter 3 we derive an exact quasilinear version of an existing exponential algorithm, allowing us to do vastly more computation with the same amount of computational resources. Finally, in Chapter 4 we show that a reasoned analysis of prior assumptions, together with a novel application of tools developed in another field, can allow us to increase the accuracy of an existing model without having to re-train. In this case the model already possessed the capabilities, we just needed to unlock them with the right viewpoint.

Our applications to deep learning are further unified by our novel viewpoint of overfitting MLP networks (see below) which can be applied to problems arising in multiple fields of study. Our work highlights the essential underlying mathematical connections between these fields and demonstrates the value in interdisciplinary applied mathematics research programs.

Analytic Shock Formation: A Jumping Off Point We begin our study by proving a result from pure mathematics regarding the construction and analysis of a class of solutions to the fractal Burgers equation which form shocks, that is they break down in finite time with a gradient singularity. In particular, we prove that starting from smooth initial data, this class of solutions converges to asymptotically self-similar, smooth, Burgers shock profiles.

This analysis is based on the methods of Buckmaster-Shkoller-Vicol developed in the papers [14,15,16, 17], and first extended to non-local problems by Yang [112]. Our work is the first to extend these styles of results to non-local dissipation, which requires careful point-wise analysis of the fractional Laplacian.

1.1. A New Perspective on Overfitting MLP Networks

Before discussing the particular applications to deep learning contained in this thesis, we present a unifying idea of a certain class of deep learning problem which is solved by what we call ”overfitting MLP networks”. Having a sense for when these problems arise will be important in informing the remainder of our study.

In this brief section we propose a novel viewpoint of certain neural network training targets which we argue puts the neural network training into the **overfitting regime**. We define this regime mathematically and argue that problems which can be posed as functional interpolation fall into this regime. As specific cases we argue that both PINN training (Chapter 3) and the interpolation of positional embeddings (Chatper 4) fall into this category. Importantly, we argue that training in this regime does not adhere to the same intuitions that standard training in machine learning does, leading us to propose counter-intuitive solutions when training networks in this regime. At a high-level, for overfitting neural networks, regularization is universally harmful, and the goal of training should be to aggressively minimize the training loss, instead of attempting to ”generalize” to a validation set.

Consider the standard supervised regression learning task: given a sample space $\mathcal{S} = \mathcal{X} \times \mathcal{Y}$ and an unknown joint probability distribution $P(x, y)$, the goal is to learn a function $h : \mathcal{X} \to \mathcal{Y}$ which is the conditional expectation $\mathbb{E}[\,Y \mid X = x\,]$ of the probability distribution $P(Y|X = x)$. Consider a dataset $\mathcal{D}$ sampled i.i.d. from $P(\mathcal{X}, \mathcal{Y})$. To learn h we aim to minimize the true risk

$$R(h) := \int_{\mathcal{S}=\mathcal{X}\times\mathcal{Y}} \mathcal{L}(h(x), y)\, dP(x, y), \tag{1.1}$$

where $\mathcal{L} \geqslant 0$ is a prescribed loss function. This is accomplished in practice by minimizing the empirical risk $E(h)$ over the finite dataset $\mathcal{D}$

$$E(h) := \frac{1}{|\mathcal{D}|} \sum_{(x,y)\in\mathcal{D}} \mathcal{L}(h(x), y). \tag{1.2}$$

Our hope is that E and R are correlated so that minimizing E has the effect of minimizing R. To minimize $E(h)$ it suffices to solve the optimization problem

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} E(h(\boldsymbol{\theta})), \tag{1.3}$$

however we are not guaranteed in general that $\boldsymbol{\theta}^* = \boldsymbol{\theta}^+$, where $\boldsymbol{\theta}^+ = \arg\min_{\boldsymbol{\theta}} R(h(\boldsymbol{\theta}))$.

We now consider a specific type of regression problem. Given an arbitrary function $f(x)$ on the smooth, bounded domain Ω, find a neural network $h(x)$ which minimizes the L^2 distance to h. I.e. train a neural network to find the solution to

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \|h(\boldsymbol{\theta}) - f\|_{L^2(\Omega)}. \tag{1.4}$$

This problem is fundamentally different from standard supervised machine learning problems which aim to produce a generalizing model from a finite dataset. The reason for this is that in present problem setting we can explicitly compute the joint probability distribution and therefore only need to directly minimize the empirical risk E.

To see why, assume that the points x are uniformly sampled from Ω. We have complete control over the sampling procedure, and could choose any sampling method we like. But for the sake of convenience it suffices to take a uniform distribution[1]. Then we have that

$$P(x, y) = \frac{1}{|\Omega|}\mathbf{1}_\Omega \cdot \delta(y - f(x)). \tag{1.5}$$

We can marginalize out the delta function appearing in the risk integral 1.1 to obtain

$$\begin{aligned} R(h) = \int_{\mathcal{S}} \mathcal{L}(h(x), y)\, dP(x, y) &= \int_{\mathcal{X}} \mathcal{L}(h(x), f(x))\, d\mu(x) \\ &= \frac{1}{|\Omega|}\int_\Omega \mathcal{L}(h(x), f(x))\, dx. \end{aligned} \tag{1.6}$$

Thus we can obtain an exact form of the risk integral which does not depend on having an estimate of the underlying data distribution. Next we will show that simple integral discritization allows us to obtain theoretical bounds on the risk integral.

Trapezoid Overfitting Inequality: The simplest example of an overfitting inequality can be derived in the case that Ω is the 1D interval $[0, 1]$. In this case we can discritize the domain with $N + 1$ samples and define the discrete domain $\widetilde{\Omega} = \{0, \Delta x, 2\Delta x, \cdots, 1\}$, where $\Delta x = 1/N$. In this case, the trapezoid rule gives

[1]Although for specific applications one may want to choose different sampling distributions and re-derive the overfitting inequalities.

the bound

$$
\begin{aligned}
R(h) &\leqslant \frac{\Delta x}{2}\sum_{k=2}^{N+1}(f(x_k)+f(x_{k-1})) + \frac{1}{12N^2}\|\mathcal{L}''\|_{L^\infty} \\
&= E(h) + \frac{|b-a|}{2N}(\mathcal{L}(x_1)+\mathcal{L}(x_{N+1})) + \frac{1}{12N^3}\|\mathcal{L}''\|_{L^\infty} \\
&\leqslant E(h) + \frac{1}{N}\sup_{x_k}\mathcal{L}(x_k) + \frac{1}{12N^3}\|\mathcal{L}''\|_{L^\infty}.
\end{aligned} \tag{1.7}
$$

Thus, the continuous risk integral 1.1 is bounded above (to leading order) by the training loss. In particular, once the number of sampling points gets sufficiently large the training loss is the only asymptotically relevant term remaining in this bound.

Monte-Carlo Overfitting Inequality: In general the Trapezoid overfitting inequality is insufficient for establishing bounds since it applies narrowly to domains in higher dimensions and scales unfavorably with high-dimensional problems. Much more general is Monte-Carlo estimation, which directly approximates the integral

Because we know a priori the underlying distribution of the points x, we can directly apply Monte-Carlo integration. Let $\widetilde{\Omega}$ consist of N points from the domain, sampled according to the underlying data distribution. Then

$$
R(h) \leqslant \frac{1}{N}\sum_{x\in\widetilde{\Omega}}\mathcal{L}(h(x), f(x)) + \mathcal{O}\left(\frac{1}{\sqrt{N}}\right) = E(h) + \mathcal{O}\left(\frac{1}{\sqrt{N}}\right). \tag{1.8}
$$

This inequality holds in any dimension and over any domain with $|\Omega| = 1$, and the error term is controlled by the variance in $\mathcal{L}$ over the domain.

1.1.1. Implications.

We discuss two salient implications which play a role generally in PINN training (Chapter 3), as well as in the interpolation which we use in Chapter 4. Other implications can be derived, but we leave this effort to future works.

Explicit Regularization is Harmful The primary implication is that in the overfitting regime, all forms of regularization will be harmful to network performance. This notably includes techniques like batch-normalization [**54**], which aim to increase the training error $E(h)$ to reduce the true risk $R(h)$, i.e. increase generalization capabilities. We see, however, from our overfitting inequalities, that techniques intended to increase $E(h)$ can be harmful to reducing the true risk.

Fourier Features Aid in Overfitting From the Trapezoid overfitting inequality (1.7) we see that higher-order derivatives of the loss affect the converging of the true risk. In cases where $\mathcal{L}''$ has high-frequency components, the 1D Sobolev inequality tells us that those high frequency components are greatly amplified

$$\|\mathcal{L}''\|_{L^\infty} \lesssim \|\mathcal{L}'''\|_{L^2} = \|\xi^3 \hat{\mathcal{L}}\|_{L^2}. \tag{1.9}$$

In particular, if the highest frequency components of the loss are on the order of $N^{3/2}$, then the error term in our overfitting inequality may dominate the empirical risk. This partially explains why neural networks are biased towards low-frequency functions [83, 111].

For PINNs in particular studies have shown empirically that Fourier features [94] greatly increase performance [75, 104, 105]. We hypothesize that this is because adding Fourier features to a network effectively reduces the high-frequency components of the loss function, and allows the empirical risk $E(h)$ to dominate the overfitting inequality (1.7). This effect is further exacerbated in PINN training, since the loss typically relies on terms which are already at least second order in space. Thus, $\mathcal{L}'' \sim f^{(4)}$. For a high-frequency target function f this means that with dominate frequencies of only $N^{3/4}$ we should already expect poor convergence.

1.2. Some Applications of Overfitting MLPs

In the final two chapters we consider two applications of overfitting MLPs and solve some problems which arise in these varied contexts. First, we look at PINNs, our study of which stems from our analytic work in Chapter 2. Next we consider a surprising application of our studies: multimodal large language models (MLLMs).

1.2.1. Physics Informed Neural Networks. Physics informed neural networks (PINNs) ae a numerical method which leverages deep learning to compute solutions to differential equations [84]. Suppose that we wish to compute the solution to the differential equation $F(\partial^\alpha u; \boldsymbol{x}) = 0$ for some multi-index α [40]. We can let $u_\theta(\boldsymbol{x})$ be a feed-forward, densely connected, neural network with parameters θ and train the neural network on the discrete domain $\Omega = \{\boldsymbol{x}_1, \cdots \boldsymbol{x}_N\}$ using the loss target

$$L(u_\theta) = \frac{1}{N}\sum_{k=1}^{N} |F(\partial^\alpha u_\theta; \boldsymbol{x}_k)|^2 + \mathrm{BC}, \tag{1.1}$$

which is the mean-squared error (MSE) of the differential equation residual with BC being appropriately enforced boundary conditions. Supposing that the solution to $F(\partial^\alpha u; \boldsymbol{x}) = 0$ lies in a space in which C^∞ functions are dense, we can theoretically train a neural network to approximate the true solution of the PDE to arbitrary precision by the approximation theorem [25, 50].

Forward PINN training of the type we study in this thesis is usually considered to be a semi-supervised learning task. However, we argue that PINN training is not only a fully supervised learning task, but that it is also an MLP overfitting problem. Indeed, the underlying data distribution can be taken to be uniform, and our training target is to minimize the residual loss (1.1). Thus, ignoring boundary conditions, PINN training exists in the overfit regime, overfitting the neural network to a function which minimizes the equation residual.

This situation is, of course, complicated by the fact that a satisfactory general theory relating the residual error to the solution error has not yet been elucidated, however see [30, 74] for some work in this direction. The overfitting neural network perspective tells us that in situations for which the residual error is dominant, we should use techniques which are expected to **reduce** generalization.

Our Contributions: During our study of PINNs we observed that training with PINNs can be prohibitively slow, but that it does not need to be. We observed that for the required computation for PINNs, the computed autograd graph is unnecessarily large. Through a straightforward argument one can see that the graph can be computed in at most (quasi) linear time, not exponential time. To this end, in Chapter 3 we extend a technique from the 1990's, developed prior to the widespread adoption of autograd, to compute the necessary computational graphs in linear, rather than exponential, runtime. We the perform extensive empirical analysis to demonstrate the practicality of our technique. We are able to compute high-order unstable self-similar Burgers profiles which were previously intractable on a single GPU.

1.2.2. General Constrained Interpolation. The overfitting MLP perspective applies in contexts beyond PINN training. We suggest that in general we have an overfitting problem whenever we have a training target for which we need to match a given function on a set of points $\mathcal{M}$ with an additional auxiliary loss term which determines the prior for interpolation. In the case we see below we must constrain our interpolation function on the fixed CLIP grid to match the fixed CLIP positional embeddings, but off of this grid we need our interpolation function to minimize the error induced by the interpolation. This auxiliary criteria has analogs the the PINN training target discussed above.

Our Contributions: In Chapter 4 we present a method for increasing the statistical efficiency of multi-modal large language models which proceeds from the observation that the CLIP vision encoder makes implicit assumptions about the visual content of images, and that these assumptions can harm the performance of the underlying LLM. We propose an alternative patchification procedure using Quadtrees, but this then introduces an issue of having to retain position information, which for CLIP is heavily dependent on the patchifcation. Thus, in able to propagate positional information downstream to the LLM from the vision encoder, we must interpolate the fixed CLIP postion embeddings. This situation is exactly the type of situation that we have described above for which overfitting MLPs are well-suited. By learning to interpolate these positional encodings we are freed up to use arbitrary patchification procedures without negatively impacting downstream model performance.

1.3. Outline

Finally, we leave the reader with an outline of what is to come. Chapter 2 presents the author's work on analytic shock formation. This work sets the stage for later chapters by providing a fertile example of a challenging mathematical problem which can be profitably explored using PINNs. Chapter 3 introduces PINNs, their relationship to partial differential equations, and the author's work on improving the efficiency of PINNs. Finally, in Chapter 4 we pursue one such application of overfitting MLP networks by using an MLP network together with a quadtree structure to improve the statistical efficiency of multimodal large-language models.

CHAPTER 2

Asymptotically Self-Similar Shock Formation for the Fractal Burgers Equation

A version of the work contained in this chapter was originally published as a paper [22].

2.1. Introduction

We will study *fractal Burgers equation*

$$
\begin{aligned}
&\partial_t u + u\partial_x u + (-\Delta)^\alpha u = 0, \qquad \text{on } \boldsymbol{R}\times\boldsymbol{R} \\
&u(x,0) = u_0(x), \qquad \text{on } \boldsymbol{R}
\end{aligned}
\tag{2.1}
$$

where the fractional Laplacian $(-\Delta)^\alpha$ is defined as the Fourier multiplier $|\xi|^{2\alpha}$, and can be represented by the Calderon-Zygmund singular integral

$$
((-\Delta)^\alpha u)(x) = C_\alpha \int_{\boldsymbol{R}} \frac{u(x)-u(\eta)}{|x-\eta|^{2\alpha+1}}\,d\eta, \qquad C_\alpha = \frac{4^\alpha \Gamma(\alpha+1/2)}{\pi^{1/2}|\Gamma(-\alpha)|},
\tag{2.2}
$$

taken in the principle value sense. For the remainder of this chapter we will assume all singular integrals are taken in a principle value sense. The formula (2.2) is well known and a derivation for any dimension is given in [27].

Equation (2.1) models the competing effects of viscous regularization and a Burgers-type nonlinearity [2, 56]. The case $\alpha = 0$ corresponds to Burger's equation, which forms shocks in finite time from smooth initial data, while $\alpha = 1$ corresponds to viscous Burger's equation, whose solutions immediately regularize to C^∞ [107]. In particular, (2.1) is a dramatically simplified model of the interaction between viscous regularization and nonlinearity in the Navier-Stokes equations.

The equation (2.1) has critical exponent $\alpha = 1/2$, at which the orders of the regularization and the nonlinearity match. It is known that equation (2.1) forms gradient discontinuities in the supercritical case $0 < \alpha < 1/2$, and remains globally smooth from smooth data in both the critical case $\alpha = 1/2$ and the subcritical case $1/2 < \alpha < 1$ [2, 56].

2.1.1. Historical Study of Fractal Burgers Equation.

Equation (2.1) has been well-studied in the last two decades. Biler, Funaki, and Woyczynski [10] established the existence of self-similar solutions, L^2 energy estimates, and global well-posedness in H^1 for the range $3/4 < \alpha \leqslant 1$, among other things. More complete studies of the existence and blowup of solutions to (2.1) were carried out independently by Alibaud, Droniou, and Vovelle [2], and Kiselev, Nazarov, and Shterenberg [56]. Although they used very different methods, both proved finite-time blowup in the supercritical case, and global existence in the critical and subcritical cases. Dong, Du, and Li [33] produced an explicit open set of initial data which form shocks in finite time. The precise relationship between the singularity time and supercritical exponents α was investigated numerically by Ramírez and Protas [85]. In the subcritical case, Akopian, Kang, and Vasseur [88] established the vanishing viscosity limit for entropy shocks. Albritton and Beekie have examined the long term asymptotic behavior of solutions to (2.1) in the critical case $\alpha = 1/2$ and showed that solutions asymptotically approach a self-similar solution [1]. In this chapter, we produce an open set of initial data which asymptotically approach a self-similar solution to Burger's equation, and characterize the Hölder regularity of the shock profile.

As this paper was being submitted to the arXiv we were made aware of an independent work by Oh and Pasqualotto which proves the same result using different techniques [77].

2.1.2. Similarity Variables, Modulation, and Singularity Formation.

The utility of similarity variables has been known for decades, and the comprehensive study of such methods was started by Barenblatt and Zel'Dovich [4, 6] in the 20th centry. Giga and Kohn [44, 45] adapted these methods for studying singularity formation in the nonlinear heat equation, and made significant contributions to our understanding of the role of self-similarity in singular behavior. Eggers and Fontelos [37, 38] have distilled the work of Barenblatt, Zel'Dovich, Giga, Kohn, and others into a digestible framework suitable for applications in many physical and analytic contexts. More recently, Collot, Ghoul, and Masmoudi [24] proved that shocks for Burgers equation are self-similar to leading order, and using similar methods, Qiu and Zhao [81] constructed solutions with simultaneous self-similar shocks at finitely many points.

The present result is inspired by the results obtained by Buckmaster, Shkoller, and Vicol in their series of papers [15, 16, 17] which prove shock formation for the 2D isentropic compressible Euler, 3D isentropic compressible Euler, and 3D non-isentropic compressible Euler equations respectively. The fundamental idea underlying the method of Buckmaster, Shkoller, and Vicol is that instabilities in the evolution correspond to symmetries of the equation under study. Informally speaking, we introduce "modulation functions" which keep track of the equivalence class of the solution as it evolves.

2.1.3. Summary of Present Result.

The present result shows that for a certain range of α, we can view the evolution (2.1) as a perturbation of *stable* Burgers shock dynamics:

THEOREM 2.1.1. ***(Imprecise Statement of Result)*** *Starting from non-degenerate initial data, the solution of* (2.1) *forms a generic point shock. Furthermore, we can compute the time, location, and regularity of this shock. Furthermore, the shock is an asymptotically self-similar Burger's shock.*

The formal statement of this result is contained in Theorem 2.2.1 and Corollary 2.2.1. Our proof closely follows the strategy laid out in the recent works [**14**, **15**, **16**, **17**, **112**], namely using modulated self-similar variables, careful bounds on Lagrangian trajectories, and a bootstrapping argument to prove the result.

Symmetry and Modulation. To illustrate the need for modulation, suppose we have an exact initial data $u(\cdot, 0)$ which we know leads to a shock, and in the course of our proof we use the fact that $u(0,0) = 0$. We want to show that in some open neighborhood of $u(\cdot, 0)$ a shock will still form. In our case, we work with the H^6 topology, in which a generic perturbation of $u(\cdot, 0)$ no longer fixes the origin.

The reader can check that Fractal Burgers equation (2.1) is invariant under the following four parameter symmetry group

$$u_*(x,t) = \frac{1}{\lambda} u\left(\frac{x - x_0}{\lambda}, \frac{t - t_0}{\lambda}\right) + u_0, \tag{2.3}$$

where u solves (2.1).

Our solution to this problem is to track certain invariants which fix the equivalence class under the relevant symmetries. For instance, when we perturb $u(\cdot, 0)$ by a sufficiently small amount to $v(\cdot, 0)$, there is a small function $\xi(t)$ such that $v(\xi(t), t) = 0$ for small t. By keeping track of ξ, we fix the equivalence class of $v(\cdot, 0)$ under spatial translation. The modulation functions fix the equivalence class of the solution and allow us to drop the odd symmetry assumptions present in [**2**] and prove shock formation on an open set in a strong topology.

Before continuing we recall some tools which will be useful in what follows.

2.1.4. Toolbox.

2.1.4.1. *A Framework for Weighted Transport Estimates.* We present a framework for doing weighted transport estimates which was originally introduced in [**15**]. See also Section 6.5 of [**112**]. We sketch the main ideas here. Our goal is to bound the L^∞ evolution of a transport-type equation by using information over some spatial region.

Consider the forced-damped transport equation

$$\partial_s \mathcal{U} + \mathcal{D}\mathcal{U} + g_W \partial_y \mathcal{U} = \mathcal{F}.$$

We denote the damping $\mathcal{D}$ and the forcing $\mathcal{F}$. The advection velocity g_W is defined in (2.5).

We consider the weighted quantity $V := \langle x\rangle^p \mathcal{U}$, where p is some power. It is straightforward to compute the evolution of V

$$\partial_s V + \left(\mathcal{D} - 2g_W p x \langle x\rangle^{-1} g_W\right) V + g_w V = \langle x\rangle^p \mathcal{F}. \tag{2.4}$$

Consider the Lagrangian trajectory Φ^{x_0} obtained above by solving (2.20), and set $V^{x_0}(s) := V \circ \Phi^{x_0}(s)$. Composing the evolution (2.4) with $\Phi^{x_0}(s)$ gives

$$\frac{d}{ds} V^{x_0} + \left(\mathcal{D} \circ \Phi^{x_0} - 2g_W p \Phi^{x_0} \langle \Phi^{x_0}\rangle^{-1} g_W\right) V^{x_0} + g_W V^{x_0} = \langle \Phi^{x_0}\rangle^p \mathcal{F} \circ \Phi^{x_0}.$$

This equation can be solved via separation of variables to obtain

$$V^{x_0}(s) = V(x_0) e^{-\int_{s_*}^{s} \mathcal{D}\circ\Phi^{x_0}(s')\, ds'} + \int_{s_*}^{s} \mathcal{F} \circ \Phi^{x_0}(s') e^{-\int_{s_*}^{s'} \mathcal{D}\circ\Phi^{x_0}(s'')\, ds''}\, ds', \tag{2.5}$$

where s_* is the first time that the trajectory enters the region in question.

2.1.4.2. *Lemmas.* We make frequent use of the following lemma to bound the fractional Laplacian in terms of our L^∞ bootstraps.

Lemma 2.1.1.1. ***(L^∞ Interpolation of the Fractional Laplacian.)*** *Let $u \in W^{1,\infty}(\boldsymbol{R})$ and $\alpha \in (0, 1/2)$, then $(-\Delta)^\alpha u \in L^\infty$ and satisfies the following estimate*

$$\|(-\Delta)^\alpha u\|_{L^\infty} \lesssim \|u\|_{L^\infty}^{1-2\alpha} \|u'\|_{L^\infty}^{2\alpha}. \tag{2.6}$$

Proof. Let $0 < h$ to be chosen later and write the fractional Laplacian in its singular integral form (2.2)

$$\begin{aligned}\frac{1}{C_\alpha}(-\Delta)^\alpha u &= \int_{\boldsymbol{R}\setminus\boldsymbol{B}_h(x)} \frac{u(x)-u(y)}{|x-y|^{2\alpha+1}}\, dy + \int_{\boldsymbol{B}_h(x)} \frac{u(x)-u(y)}{|x-y|^{2\alpha+1}}\, dy \\ &=: I_1 + I_2.\end{aligned}$$

For the first integral we estimate

$$\begin{aligned}|I_1| \leqslant \int_{\boldsymbol{R}\setminus\boldsymbol{B}_h(x)} \frac{u(x)-u(y)}{|x-y|^{1+2\alpha}}\, dy \leqslant 4\|u\|_{L^\infty} \int_h^\infty \frac{1}{y^{1+2\alpha}}\, dy \\ \leqslant \frac{1}{\alpha h^{2\alpha}} \|u\|_{L^\infty}.\end{aligned}$$

For the second integral, the mean value theorem gives

$$u'(\eta,t) = \frac{u(x,t)-u(y,t)}{x-y}, \qquad |\eta| \leqslant |x-y|.$$

This implies that

$$\begin{aligned}
|I_2| \leqslant \int_{\boldsymbol{B}_h(x)} \frac{u(x)-u(y)}{|x-y|^{1+2\alpha}}\,dy &= \int_{\boldsymbol{B}_h(x)} \frac{u'(\eta)}{|x-y|^{2\alpha}}\,dy \\
&\leqslant 2\|u'\|_{L^\infty} \int_0^h \frac{1}{y^{2\alpha}}\,dy \\
&= \frac{1}{1-2\alpha} h^{1-2\alpha} \|u'\|_{L^\infty}.
\end{aligned}$$

To summarize, we now have

$$\frac{1}{C_\alpha}(-\Delta)^\alpha u \leqslant \frac{1}{\alpha h^{2\alpha}}\|u\|_{L^\infty} + \frac{1}{1-2\alpha}h^{1-2\alpha}\|u'\|_{L^\infty}.$$

Letting $A = \|u\|_{L^\infty}$, $B = \|u'\|_{L^\infty}$, and $h = A^\beta B^\gamma$, we see that the only possible scaling is $\beta = 1, \gamma = -1$, i.e. we choose $h = \|u\|_{L^\infty}/\|u'\|_{L^\infty}$ and obtain

$$\frac{1}{C_\alpha}(-\Delta)^\alpha u \leqslant \left(\frac{1}{\alpha} + \frac{1}{1-2\alpha}\right)\|u\|_{L^\infty}^{1-2\alpha}\|u'\|_{L^\infty}^{2\alpha}.$$

■

We also regularly use the Gagliardo-Nirenberg Interpolation Theorem for $\boldsymbol{R}$.

Lemma 2.1.1.2. ***(Gagliardo-Nirenberg Interpolation).*** *Fix $1 \leqslant q, r \leqslant \infty$, $j, m \in \boldsymbol{N}$ with $j/m \leqslant \theta \leqslant 1$. If we have the following relationship*

$$\frac{1}{p} = j + \theta\left(\frac{1}{r} - m\right) + \frac{1-\theta}{q},$$

then

$$\|\partial_x^j u\|_{L^p} \lesssim \|\partial_x^m u\|_{L^r}^\theta \|u\|_{L^q}^{1-\theta}.$$

The implicit constant depends only on j, m, r, p, q, and θ (i.e. is independent of u).

2.1.4.3. *Derivation and Properties of the Self-Similar Burgers Profile Family.* We make frequent use of the self-similar Burgers profile in what follows. Consider the similarity equation for the Burgers equation [38]

$$-\frac{1}{2}\Psi + \left(\frac{3}{2}y + \Psi\right)\Psi' = 0. \tag{2.7}$$

This is a homogeneous ODE in Ψ. It is elementary to obtain the following implicit family of solutions

$$x = -\Psi_\nu - \frac{\nu}{6}\Psi_\nu^3, \tag{2.8}$$

where $\nu/6$ is the constant of integration. We denote by Ψ_ν the solution of (2.7) corresponding to a particular choice of ν.

Taking successive (implicit) derivatives we easily obtain the following properties of Ψ_ν:

$$\Psi_\nu(0) = 0, \quad \Psi_\nu'(0) = -1, \quad \Psi_\nu''(0) = 0, \quad \Psi_\nu'''(0) = \nu, \quad \Psi_\nu^{(2k)}(0) = 0, \quad k \in \boldsymbol{N}. \tag{2.9}$$

This shows that the solution family (2.8) is parameterized by it's third derivative at the origin.

We take by convention $\Psi := \Psi_6$, which has the explicit solution

$$\Psi(x) = \left(-\frac{x}{2} + \left(\frac{1}{27} + \frac{x^2}{4}\right)^{1/2}\right)^{1/3} - \left(\frac{x}{2} + \left(\frac{1}{27} + \frac{x^2}{4}\right)^{1/2}\right)^{1/3}. \tag{2.10}$$

We use this particular profile throughout our analysis. Furthermore we can Taylor expand the derivative Ψ' around $x = 0$ to obtain

$$\Psi'(x) = -1 + 3x^2 - 15x^4 + \mathcal{O}(x^6). \tag{2.11}$$

In fact, once we have obtained the solution for Ψ, the solutions for all of the profiles are obtained through the formula

$$\Psi_\nu(x) = \left(\frac{\nu}{6}\right)^{-1/2} \Psi\left(\left(\frac{\nu}{6}\right)^{1/2} x\right). \tag{2.12}$$

This can be verified by noting that if Ψ solves (2.8) for $\nu = 1$, then Ψ_ν solves the same equation with, excusing the abuse of notation, $\nu = \nu$.

We note that the explicit form of *any* of the solutions in the family (2.8) can be obtained quite easily using Mathematica or some other solver (or by hand if the reader is a spiritual medium for Gerolamao Cardano), since the explicit solution (2.10) is nothing more than the solution of the cubic $x = -y - \nu y^3$.

Once an explicit solution is obtained, the reader can easily verify the following inequalities which (we suggest using Mathematica)

$$|\Psi(x)| \leqslant |x|^{1/3}, \tag{2.13}$$

along with the Japanese bracket estimates

$$|\partial_x^i \Psi(x)| \lesssim \langle x\rangle^{1/3-i}, \tag{2.14}$$

for $i = 1, ..., 5$ and $x > 1$. In the case of $i = 1, 2$ the constant can be taken to be 1. The constant for $i = 5$, for example, can be taken as 360.

The fractional Laplacian applied to the stable profile $\Psi^{(n)}$ behaves exactly as one would expect, namely we have the following pointwise estimate:

Lemma 2.1.1.3. *The bounds* (2.14) *imply*

$$|(-\Delta)^\alpha \Psi'(x)| \lesssim \langle x\rangle^{-2/3-2\alpha}. \tag{2.15}$$

Proof. Without loss of generality, suppose $x > 0$. We decompose the integral form (2.2) of $(-\Delta)^\alpha$ by

$$(-\Delta)^\alpha \Psi'(x) = C_\alpha \left(\int_{-\infty}^{-x} + \int_{-x}^{x/2} + \int_{x/2}^{x} + \int_{x}^{2x} + \int_{2x}^{\infty} \right) \frac{\Psi'(x) - \Psi'(y)}{|x-y|^{1+2\alpha}}\, dy.$$

The first integral can be bounded as

$$\int_{-\infty}^{-x} \frac{|\Psi'(x) - \Psi'(y)|}{|x-y|^{1+2\alpha}}\, dy \lesssim \langle x\rangle^{-2/3} \int_{-\infty}^{-x} \frac{1}{|x-y|^{1+2\alpha}}\, dy \lesssim \langle x\rangle^{-2/3-2\alpha}.$$

The second integral can be bounded with

$$\int_{-x}^{x/2} \frac{|\Psi'(x) - \Psi'(y)|}{|x-y|^{1+2\alpha}}\, dy \lesssim \int_{-x}^{x/2} \frac{x^{-2/3} + |y|^{-2/3}}{x^{1+2\alpha}}\, dy \lesssim x^{-2/3-2\alpha} + \frac{1}{x^{1+2\alpha}} \int_{-x}^{x/2} y^{-2/3}\, dy \lesssim \langle x\rangle^{-2/3-2\alpha}.$$

To bound the third integral, we use the Holder seminorm bound

$$|\Psi'|_{C^{3\alpha}(x/2,x)} \leqslant \|\Psi'\|_{L^\infty(x/2,x)}^{1-3\alpha} \|\Psi''\|_{L^\infty(x/2,x)}^{3\alpha} \lesssim (\langle x\rangle^{-2/3})^{1-3\alpha} (\langle x\rangle^{-5/3})^{3\alpha} = \langle x\rangle^{-2/3-3\alpha}.$$

Using this seminorm bound, we obtain that the third integral can be estimated as

$$\lesssim |\Psi'|_{C^{3\alpha}(x/2,x)} \int_{x/2}^{x} \frac{1}{|x-y|^{1-\alpha}}\, dy \lesssim \langle x\rangle^{-2/3-3\alpha} x^\alpha \lesssim \langle x\rangle^{-2/3-2\alpha}.$$

The fourth integral is estimated similarly to the second integral, and the fifth integral is estimated similarly to the first. ■

2.2. Self-Similar Transformation, Initial Data, and Statement of Theorem

Our solutions will develop a gradient discontinuity at an *a priori* unknown time T_* and location x_*. Throughout ε denotes a small parameter which will be fixed during our proof.

2.2.1. The Self-Similar Transformation. We define the modulation variables $\tau, \xi, \kappa : [-\varepsilon, T_*] \to \boldsymbol{R}$, which respectively control the temporal location, spatial location, and density of the developing shock. We fix their initial values at time $t = -\varepsilon$ to be

$$\tau(-\varepsilon) = 0, \quad \xi(-\varepsilon) = 0, \quad \kappa(-\varepsilon) = \kappa_0. \tag{2.1}$$

By their definition, the modulation variables will satisfy $\tau(T_*) = T_*$ and $\xi(T_*) = x_*$, meaning that the gradient blowup occurs at the unique fixed point of τ.

We make the change to *modulated self-similar* variables described by

$$y(x,t) = \frac{x - \xi(t)}{(\tau(t) - t)^{3/2}}, \qquad s(t) = -\log(\tau(t) - t) \tag{2.2}$$

and the modulated self-similar variables ansatz

$$u(x,t) = e^{-\frac{s}{2}} W(y,s) + \kappa(t). \tag{2.3}$$

This ansatz corresponds to the stable self-similar scaling for the stable Burgers profile Ψ (c.f. Section 2.1.4.3). This scaling is a natural choice since we are considering the dynamics of (2.1) as a perturbation from the stable Burgers dynamics.

Plugging the ansatz (2.3) into (2.1) we obtain the PDE

$$\left(\partial_s - \frac{1}{2}\right) W + \left(\beta_\tau (W + e^{\frac{s}{2}}(\kappa - \dot{\xi})) + \frac{3}{2} y\right) \partial_x W = -\beta_\tau e^{-\frac{s}{2}} \dot{\kappa} - \beta_\tau e^{(3\alpha - 1)s} (-\Delta)^\alpha W \tag{2.4}$$

for W in y and s and where

$$\beta_\tau := \frac{1}{1 - \dot{\tau}}.$$

For convenience we define the transport speed

$$g_W := \beta_\tau \left(W + e^{\frac{s}{2}}(\kappa - \dot{\xi})\right) + \frac{3}{2} y. \tag{2.5}$$

The equations governing the evolution of the derivatives of W are recorded in Section 2.2.3.

2.2.1.1. *Modulation Functions and Constraints on the Evolution.* Imposing the following constraints at $y = 0$ fully characterizes the developing shock, as well as fixing our choice of τ, ξ and κ:

$$W(0,s) = 0, \quad \partial_y W(0,s) = -1, \quad \partial_y^2 W(0,s) = 0. \tag{2.6}$$

For any function φ we denote $\varphi(0,s) =: \varphi^0(s) = \varphi^0$ out of convenience. We record the following identities for τ and ξ

$$\dot{\tau} = -e^{(3\alpha-1)s}\left((-\Delta)^\alpha \partial_y W\right)^0(s), \tag{2.7a}$$

$$\kappa - \dot{\xi} = -e^{-s}\dot{\kappa} - e^{(3\alpha-\frac{3}{2})s}\left((-\Delta)^\alpha W\right)^0(s) = -\frac{e^{(3\alpha-\frac{3}{2})s}}{\partial_y^3 W^0(s)}\left((-\Delta)^\alpha \partial_y^2 W\right)^0(s). \tag{2.7b}$$

We compute (2.7a) by evaluating (2.22) at $y = 0$. Equation (2.7b) follows from evaluating (2.23) and (2.4) at $y = 0$.

2.2.2. Assumptions on the Initial Data. Let $M > 1$ be large, $\varepsilon < 1$ small. These constants will be fixed in the course of the proof and are independent of one another. This ensures that that we can make quantities of the form $M^p\varepsilon$, $p > 0$, arbitrarily small by taking ε sufficiently small. We use this frequently in what follows. We define the constants:

$$m = \frac{3}{16}(1-3\alpha), \qquad \ell = \frac{7}{8}(1-3\alpha) \qquad q = \min\{\frac{2}{3}(1-3\alpha), \frac{1}{6}\}, \tag{2.8}$$

which are chosen to satisfy certain constraints arising during the course of the bootstrap argument. We use the notation $\langle x\rangle := (1+x^2)^{1/2}$ throughout and set

$$s_0 = -\log(\varepsilon) \qquad \text{and} \qquad h = (\log M)^{-2}. \tag{2.9}$$

Note that h depends on M but not on ε.

We break $\boldsymbol{R}$ into three regions. We call $\boldsymbol{B}_h(0)$ the *Taylor region* since we will rely on Taylor expanding to close our bootstraps here. The next region is the time dependent annulus $\boldsymbol{B}_{e^{ms}}(0) \setminus \boldsymbol{B}_h(0)$ with m defined in (2.8). Finally we have the *far field* which is simply $\boldsymbol{R} \setminus \boldsymbol{B}_{e^{ms}}(0)$. Since the fractal Burgers equation does not propagate compact support, we must maintain quantitative control on the solution in this region.

2.2.2.1. *Initial Data in Self-Similar Variables.* We choose initial data $W(0,s)$ in self-similar variables satisfying the following pointwise equality at the origin:

$$W^0(s_0) = 0, \qquad \partial_y W^0(s_0) = -\|\partial_y W(\cdot, s_0)\|_{L^\infty} = -1, \qquad \partial_y W^0(s_0) = 0. \tag{2.10}$$

We define $\widetilde{W} := W - \Psi$, where Ψ is the exact, self-similar Burgers profile (2.10). In this notation we assume

$$|\partial_y^3 \widetilde{W}^0(s_0)| \leqslant 6\varepsilon^{\frac{16}{9}(1-3\alpha)}, \tag{2.11}$$

$$|\widetilde{W}(y,s_0)| \leqslant \begin{cases} \varepsilon^{1-3\alpha}, & 0 \leqslant |y| \leqslant h \quad \text{(2.12a)} \\ \dfrac{1}{2}\varepsilon^{q}\langle y\rangle^{1/3}, & h \leqslant |y| < \infty, \quad \text{(2.12b)} \end{cases}$$

and

$$|\partial_y \widetilde{W}(y,s_0)| \leqslant \begin{cases} \varepsilon^{1-3\alpha}, & 0 \leqslant |y| \leqslant h \quad \text{(2.13a)} \\ \dfrac{3}{4}\varepsilon^{\frac{3}{4}\ell}\langle y\rangle^{-2/3}, & h \leqslant |y| \leqslant \varepsilon^{-m} \quad \text{(2.13b)} \\ |\partial_y \widetilde{W}(y,s_0)| \leqslant 2\varepsilon^2, & \varepsilon^{-m} \leqslant |y|. \quad \text{(2.13c)} \end{cases}$$

as well as the following bounds on the higher derivatives in the Taylor region $0 \leqslant |y| \leqslant h$

$$|\partial_y^2 \widetilde{W}|(y,s_0) \leqslant \varepsilon^{1-3\alpha}, \qquad |\partial_y^3 \widetilde{W}|(y,s_0) \leqslant \varepsilon^{1-3\alpha}, \qquad |\partial_y^4 \widetilde{W}|(y,s_0) \leqslant \varepsilon^{1-3\alpha}. \tag{2.14}$$

We will assume the norm bounds

$$\|\partial_y W(\cdot,s_0)\|_{L^2} \leqslant 200, \qquad \|\partial_y^3 W(\cdot,s_0)\|_{L^\infty} \leqslant \frac{M^{\frac{4}{7}}}{2}, \qquad \|\partial_y^6 W(\cdot,s_0)\|_{L^2} \leqslant \frac{M^{1/2}}{2}. \tag{2.15}$$

2.2.2.2. *Initial Data in Physical Variables.* We translate (2.10)-(2.15) into the physical variable. Our pointwise equalities at the origin become

$$u_0(0) = \kappa_0, \qquad u_0'(0) = -\varepsilon^{-1} = -\|u_0'\|_{L^\infty}, \qquad u_0''(0) = 0. \tag{2.16}$$

Inequality (2.11) becomes

$$|\partial_\theta^3 u_0(0) - 6\varepsilon^{-4}| \leqslant 6\varepsilon^{-4/9(5+12\alpha)}. \tag{2.17}$$

The initial bounds (2.12)-(2.13) in the self-similar variables become

$$|u(x,-\varepsilon)| \leqslant \begin{cases} \varepsilon^{\frac{1}{2}}\left(\Psi(x\varepsilon^{-\frac{3}{2}}) + \varepsilon^{1+3\alpha}\right) + \kappa_0, & 0 \leqslant |x| \leqslant h\varepsilon^{\frac{3}{2}} \quad \text{(2.18a)} \\ \varepsilon^{\frac{1}{2}}\left(\Psi(x\varepsilon^{-\frac{3}{2}}) + \dfrac{1}{2}\varepsilon^{q}\langle x\varepsilon^{-\frac{3}{2}}\rangle^{1/3}\right) + \kappa_0, & h\varepsilon^{\frac{3}{2}} \leqslant |x| < \infty \quad \text{(2.18b)} \end{cases}$$

$$|\partial_x u(x,-\varepsilon)| \leqslant \begin{cases} \varepsilon^{-1}\left(\Psi'(x\varepsilon^{-\frac{3}{2}}) + \varepsilon^{1-3\alpha}\right), & 0 \leqslant |x| \leqslant h\varepsilon^{\frac{3}{2}} \quad (2.19a) \\ \varepsilon^{-1}\left(\Psi'(x\varepsilon^{-\frac{3}{2}}) + \frac{3}{4}\varepsilon^{\frac{3}{4}\ell}\langle x\varepsilon^{-\frac{3}{2}}\rangle^{-2/3}\right), & h\varepsilon^{\frac{3}{2}} \leqslant |x| \leqslant \varepsilon^{-m+\frac{3}{2}} \quad (2.19b) \\ \varepsilon^{-1}\left(\Psi'(x\varepsilon^{-\frac{3}{2}}) + \frac{1}{2}\varepsilon^{1/2}2\varepsilon\right), & \varepsilon^{-m+\frac{3}{2}} \leqslant |x| < \infty \quad (2.19c) \end{cases}$$

The higher derivative bounds (2.14) translate to

$$\begin{aligned} \partial_x^2 u(x,-\varepsilon) &\leqslant \varepsilon^{-5/2}\left(\varepsilon^{1-3\alpha} + \Psi^{(2)}(x\varepsilon^{-\frac{3}{2}})\right), \\ \partial_x^3 u(x,-\varepsilon) &\leqslant \varepsilon^{-4}\left(\varepsilon^{1-3\alpha} + \Psi^{(3)}(x\varepsilon^{-\frac{3}{2}})\right), \\ \partial_x^4 u(x,-\varepsilon) &\leqslant \varepsilon^{-11/2}\left(\varepsilon^{1-3\alpha} + \Psi^{(4)}(x\varepsilon^{-\frac{3}{2}})\right), \end{aligned} \tag{2.20}$$

which are valid in the region $0 \leqslant |x|x \leqslant h\varepsilon^{\frac{3}{2}}$. The norm bounds (2.15) become

$$\|\partial_x u(\cdot,-\varepsilon)\|_{L^2}^2 = \varepsilon^{-\frac{1}{2}}\|\partial_x W(\cdot,s_0)\|_{L^2}^2 \leqslant 40000\,\varepsilon^{-\frac{1}{2}}, \tag{2.21a}$$

$$\|\partial_x^3 u(\cdot,-\varepsilon)\|_{L^\infty} \leqslant \varepsilon^{-4}\|\partial_x^3 W_0\|_{L^\infty} = \frac{\varepsilon^{-4}}{2}M^{\frac{4}{7}}, \tag{2.21b}$$

$$\|\partial_x^6 u(\cdot,-\varepsilon)\|_{L^2} = \varepsilon^{-4}\|\partial_x^6 W_0\|_{L^2} \leqslant \frac{\varepsilon^{-\frac{31}{4}}}{2}M^{1/2}. \tag{2.21c}$$

2.2.3. Evolution Equations For Derivatives and Differences. Our analysis requires the specific differentiated forms of (2.4) to proceed, which we record here. The equations are written in the form $(\partial_s + D)f + g_W f_y = Ff$ where D is the damping and F the forcing.

We have the following differentiated forms of equation (2.4)

$$\left(\partial_s + 1 + \beta_\tau \partial_y W\right)\partial_y W + g_W \partial_y^2 W = -\beta_\tau e^{(3\alpha-1)s}(-\Delta)^\alpha \partial_y W \tag{2.22}$$

$$\left(\partial_s + \frac{5}{2} + 3\beta_\tau \partial_y W\right)\partial_y^2 W + g_W \partial_y^3 W = -\beta_\tau e^{(3\alpha-1)s}(-\Delta)^\alpha \partial_y^2 W \tag{2.23}$$

$$\left(\partial_s + 4 + 4\beta_\tau \partial_y W\right)\partial_y^3 W + g_W \partial_y^4 W = -\beta_\tau\left[e^{(3\alpha-1)s}(-\Delta)^\alpha \partial_y^3 W + 3(\partial_y^2 W)^2\right] \tag{2.24}$$

$$\left(\partial_s + \frac{11}{2} + 7\beta_\tau \partial_y W\right)\partial_y^6 W + g_W \partial_y^7 W = -\beta_\tau\left[e^{(3\alpha-1)s}(-\Delta)^\alpha \partial_y^6 W + 35\partial_y^3 W \partial_y^4 W + 21\partial_y^2 W \partial_y^5 W\right] \tag{2.25}$$

And the equations for the differences $\widetilde{W}$, $\partial_y\widetilde{W}$, and $\partial_y^4\widetilde{W}$ derivative

$$\left(\partial_s - \frac{1}{2} + \beta_\tau\Psi\right)\widetilde{W} + g_W\partial_y\widetilde{W} = -\beta_\tau\left[e^{-\frac{s}{2}}\dot{\kappa} + e^{(3\alpha-1)s}(-\Delta)^\alpha W + \Psi'(\dot{\tau}\Psi + e^{\frac{s}{2}}(\kappa - \dot{\xi})\right] \tag{2.26}$$

$$\begin{aligned}&\left(\partial_s + 1 + \beta_\tau\left(\partial_y\widetilde{W} + 2\Psi'\right)\right)\partial_y\widetilde{W} + g_W\partial_y^2\widetilde{W}\\ &= -\beta_\tau\left[e^{(3\alpha-1)s}(-\Delta)^\alpha\partial_y W + \left(e^{s/2}(\kappa-\dot{\xi}) + \widetilde{W} + \dot{\tau}\Psi\right)\Psi'' + \dot{\tau}(\Psi')^2\right]\end{aligned} \tag{2.27}$$

$$\begin{aligned}\left(\partial_s + \frac{11}{2} + 5\beta_\tau\partial_y W\right)\partial_y^4\widetilde{W} + g_W\partial_y^4\widetilde{W} = -\beta_\tau\Big[&e^{(3\alpha-1)s}(-\Delta)^\alpha\partial_y^4 W + \left(e^{s/2}(\kappa-\dot{\xi}) + \widetilde{W} + \dot{\tau}\Psi\right)\partial_y^5\Psi\\ &+ 4\partial_y\widetilde{W}\partial_y^4\Psi + 8\partial_y^2\widetilde{W}\partial_y^3\Psi + 10\partial_y^3\widetilde{W}\partial_y^2\Psi\\ &+11\partial_y^2\widetilde{W}\partial_y^3\widetilde{W} + 5\dot{\tau}\partial_y\Psi\partial_y^4\Psi + 10\dot{\tau}\partial_y^2\Psi\partial_y^3\Psi\Big]\end{aligned} \tag{2.28}$$

2.2.4. Main Theorem. We are now equipped to state the main result of this work

Theorem 2.2.1. *There exist constants $M, \varepsilon > 0$ sufficiently large and small respectively such that for some $u_0 \in H^6(\boldsymbol{R})$, with compact support, satisfying (2.18)-(2.21a) at $t = -\varepsilon$, the unique solution to (2.1) forms a shock in finite time. Furthermore*

(i) For any $\overline{T} < T_*$, $u \in C([-\varepsilon, \overline{T}]; C^5(\boldsymbol{R}))$.

(ii) The blowup location $\xi(T_*) = x_*$ is unique.

(iii) The blowup time T_* occurs prior to $t = \frac{3}{4}\varepsilon^{\frac{8}{9}(1-3\alpha)}$ and the blowup location x_* satisfies $|x_*| \leqslant 4M\varepsilon$. The blowup time and location are explicitly computable.

(iv) $\lim_{t\to T_*}\partial_\theta u = -\infty$ and furthermore we have the precise control

$$\frac{1}{2(T_*-t)} \leqslant |\partial_\theta u(\xi(t),t)| = \|\partial_\theta u\|_{L^\infty} \leqslant \frac{1}{T_*-t}.$$

(v) The solution u converges asymptotically in the self-similar space to a self-similar solution to the Burgers equation Ψ_ν given by (2.12). I.e.

$$\limsup_{s\to\infty}\|W - \Psi_\nu\|_{L^\infty} = 0.$$

(vi) The solution has Hölder regularity $1/3$ at the blowup time, i.e. $u(x, T_*) \in C^{1/3}(\boldsymbol{R})$.

An example of an initial data satisfying these conditions and leading to a shock in finite time is $\varepsilon^{1/2}\eta(x)\Psi(x\varepsilon^{-3/2})$ for a suitable compactly supported smooth function η.

The gradient blowup outlined in Theorem 2.2.1 is stable and we can relax the initial conditions above and still satisfy the conclusions of Theorem 2.2.1:

Corollary 2.2.1. *There exists an open set of data in the H^6 topology such that the conclusions of Theorem 2.2.1 still hold.*

We prove Theorem (2.2.1) in section 2.5.3 and the corollary 2.2.1 in section 2.5.4.

2.3. Bootstraps: Assumptions and Consequences

Throughout we use $\lesssim$ to denote a quantity which is bounded by a constant depending on α or h, but not on s, ε, or M. We establish our bootstrap assumptions in 2.3.1, derive consequences from these assumptions in 2.3.2, close our top order energy estimate in 2.3.3, and establish control on the Lagrangian trajectories in 2.3.4. In Section 2.4 we close our bootstraps in L^∞.

2.3.1. Bootstrap Assumptions.

Recall that $\widetilde{W} = W - \Psi$, where Φ is the stable Burgers profile (2.10). We assume the bootstraps

$$|\widetilde{W}|(y,s) \leqslant \begin{cases} \left(\varepsilon^{\frac{8}{9}(1-3\alpha)} + \log M \varepsilon^{\frac{7}{9}(1-3\alpha)}\right) h^4, & 0 \leqslant |y| \leqslant h \quad (2.1a) \\ \varepsilon^q \langle y \rangle^{1/3}, & h \leqslant |y| < \infty \quad (2.1b) \end{cases}$$

$$|\partial_y \widetilde{W}|(y,s) \leqslant \begin{cases} \left(\varepsilon^{\frac{8}{9}(1-3\alpha)} + \log M \varepsilon^{\frac{7}{9}(1-3\alpha)}\right) h^3, & 0 \leqslant |y| \leqslant h \quad (2.2a) \\ \varepsilon^{\frac{3}{4}\ell} \langle y \rangle^{-2/3}, & h \leqslant |y| \leqslant e^{ms} \quad (2.2b) \\ 2e^{-s}, & e^{ms} \leqslant |y| \quad (2.2c) \end{cases}$$

on the solution and the first derivative. For the higher derivatives we assume the following bootstraps on the Taylor region $0 \leqslant |y| \leqslant h$

$$|\partial_y^2 \widetilde{W}|(y,s) \leqslant \left(\varepsilon^{\frac{8}{9}(1-3\alpha)} + \log M \varepsilon^{\frac{7}{9}(1-3\alpha)}\right) h^2, \tag{2.3a}$$

$$|\partial_y^3 \widetilde{W}|(y,s) \leqslant \left(\varepsilon^{\frac{8}{9}(1-3\alpha)} + \log M \varepsilon^{\frac{7}{9}(1-3\alpha)}\right) h, \tag{2.3b}$$

$$|\partial_y^4 \widetilde{W}|(y,s) \leqslant \varepsilon^{\frac{7}{9}(1-3\alpha)}, \tag{2.3c}$$

for h given by (2.9) and m, ℓ, q given by (2.8).

To compensate for a loss of derivatives, we assume the following L^∞ control over the third derivative

$$\|\partial_y^3 W\|_{L^\infty} \leqslant M. \tag{2.4}$$

Finally, we assume the precise bootstrap

$$|\partial_y^3 W(0,s) - 6| \leqslant 10\varepsilon^{\frac{16}{9}(1-3\alpha)} < 1 \tag{2.5}$$

which will ensure that our solution converges to the correct profile. For the modulation variables we assume

$$|\tau(t)| \leqslant \varepsilon^{\frac{8}{9}(1-3\alpha)}, \quad |\xi(t)| \leqslant 4M\varepsilon, \tag{2.6a}$$

$$|\dot{\tau}(t)| \leqslant \varepsilon^{\frac{8}{9}(1-3\alpha)}, \quad |\dot{\xi}(t)| \leqslant 5M \tag{2.6b}$$

for all $t < T_*$.

2.3.2. Consequences of Bootstraps. The following are immediate consequences of the bootstraps (2.1)-(2.6).

(1) The explicit property (2.14) of Ψ' along with (2.2) shows that $\partial_y W \in L^\infty_{y,s}(\boldsymbol{R})$ with the following uniform bound

$$\|\partial_y W\|_{L^\infty_{y,s}} = 2. \tag{2.7}$$

(2) From the bootstrap (2.6b) on $\dot{\tau}$ we have the following bound on β_τ

$$\beta_\tau = \frac{1}{1-\dot{\tau}} \leqslant \frac{1}{1-\varepsilon^{\frac{1}{2}(1-3\alpha)}} \leqslant 1 + 2\varepsilon^{\frac{1}{2}(1-3\alpha)} \leqslant \frac{3}{2}. \tag{2.8}$$

This bound is true for $0 \leqslant \varepsilon < \frac{1}{8}$, and we will end up taking ε much smaller than this.

(3) We have the following bound on κ:

$$|\kappa(t)| = |u(\xi(t),t)| \leqslant \|u\|_{L^\infty_{x,t}} \leqslant M. \tag{2.9}$$

This is shown with the following argument. Equation (2.1) satisfies a trivial maximum principle: if $\|u_0\|_{L^\infty} \leqslant M$ than $\|u\|_{L^\infty_{\theta,t}} \leqslant M$. This is an easy consequence of $(-\Delta)^\alpha u(x_0,t_0)$ being positive definite at a maximum. Then observe that $\kappa(t) = u(\xi(t),t)$ by (2.3) and the constraint (2.6).

(4) The transformation (2.3) and the bound (2.9) on κ give

$$\|W\|_{L^\infty_y} \leqslant e^{s/2}(\|u_0\|_{L^\infty} + M). \tag{2.10}$$

The bootstrap (2.2) is insufficient to guarantee that $\partial_y W \in L^2$; we must use the structure of the equation (2.4).

LEMMA 2.3.0.1. ***(Uniform L^2 Bound for $\partial_y W$).*** *The bootstraps (2.1)-(2.2c) imply that $\partial_y W$ is uniformly $L^2(\boldsymbol{R})$ in self-similar time.*

PROOF. $\partial_y W$ solves (2.22), which we recall for convenience

$$\Big(\partial_s + 1 + \beta_\tau \partial_y W\Big)\partial_y W + g_W \partial_y^2 W = -\beta_\tau e^{(3\alpha-1)s}(-\Delta)^\alpha \partial_y W.$$

Taking the L^2 inner product with (2.22) gives

$$\begin{aligned}\frac{1}{2}\frac{d}{ds}\|\partial_y W\|_{L^2}^2 + \|\partial_y W\|_{L^2}^2 + \int_{\boldsymbol{R}} g_W \partial_y W \partial_y^2 W\, dy + \beta_\tau \int_{\boldsymbol{R}} (\partial_y W)^3\, dy = \\ - \beta_\tau e^{(3\alpha-1)s}((-\Delta)^\alpha \partial_y W, \partial_y W)_{L^2}.\end{aligned} \tag{2.11}$$

$(-\Delta)^\alpha$ is self-adjoint on L^2 and thus we have $((-\Delta)^\alpha \partial_y W, \partial_y W)_{L^2} \geqslant 0$. The transport term satisfies

$$\begin{aligned}\int_{\boldsymbol{R}} g_w \partial_y W \partial_y^2 W\, dy &= -\frac{1}{2}\int_{\boldsymbol{R}} g_W \frac{d}{dy}(\partial_y W)^2\, dy \\ &= -\frac{\beta_\tau}{2}\int_{\boldsymbol{R}} (\partial_y W)^3\, dy - \frac{3}{4}\|\partial_y W\|_{L^2}^2.\end{aligned} \tag{2.12}$$

From the uniform L^∞ bound (2.7) and the bootstraps (2.2b), (2.2c) it follows that

$$\begin{aligned}\int_{\boldsymbol{R}} (\partial_y W)^3\, dy &= \left(\int_{0\leqslant|y|\leqslant h} + \int_{h\leqslant|y|\leqslant e^{ms}} + \int_{e^{ms}\leqslant|y|<\infty}\right)(\partial_y W)^3\, dy \\ &\leqslant C + \varepsilon^{\frac{9}{4}\ell}\int_{h\leqslant|y|\leqslant e^{ms}} \frac{1}{1+y^2}\, dy + 2\int_{e^{ms}\leqslant|y|<\infty} e^{-s}|\partial_y W|^2\, dy \\ &\leqslant C + 2\arctan(e^{ms}) + 2e^{-s}\|\partial_y^2 W\|_{L^2}^2 \\ &\leqslant C + 2\varepsilon\|\partial_y W\|_{L^2}^2\end{aligned} \tag{2.13}$$

where C is a constant which changes from line to line.

Combining (2.11) with (2.12)-(2.13) gives us

$$\begin{aligned}\frac{d}{ds}\|\partial_x W\|_{L^2}^2 &= \frac{C\beta_\tau}{2} + \left(\beta_\tau 2\varepsilon^b - \frac{1}{4}\right)\|\partial_x W\|_{L^2}^2 \\ &\leqslant \frac{C\beta_\tau}{2} - \frac{1}{8}\|\partial_x W\|_{L^2}^2\end{aligned}$$

where the last inequality is obtained by taking ε sufficiently small and estimating β_τ using (2.8). Use Grönwall's inequality with the initial data assumption (2.15) to conclude. ■

2.3.3. Closure of Top Order Energy Estimate. The equation (2.1) suffers an L^∞ loss of derivatives; we cannot estimate $\partial_x^k u$ in L^∞ without control of $\partial_x^{k+1} u$ in L^∞. To overcome this difficulty we use our bootstraps to linearize the evolution of the sixth derivative in L^2 and close the bootstrap (2.4) on the third derivative by interpolation (see Lemma 2.1.1.2).

The evolution of $\partial_y^6 W$ is given by (2.25) which we reproduce for convenience

$$\left(\partial_s + \frac{11}{2} + 7\beta_\tau \partial_y W\right)\partial_y^6 W + g_W \partial_y^7 W = -\beta_\tau \left[e^{(3\alpha-1)s}(-\Delta)^\alpha \partial_y^6 W + 35\partial_y^3 W \partial_y^4 W + 21 \partial_y^2 W \partial_y^5 W\right].$$

We test in L^2 against $\partial_y^6 W$

$$\begin{aligned}\frac{1}{2}\frac{d}{ds}\|\partial_y^6 W\|_{L^2}^2 + \frac{17}{2}\|\partial_y^6 W\|_{L^2}^2 = &-\underbrace{\int_{\boldsymbol{R}} g_W \partial_y^7 W \partial_y^6 W \, dy}_{I} - 7\beta_\tau \int_{\boldsymbol{R}} \partial_y W (\partial_y^6 W)^2 \, dy \\ &- 35\beta_\tau \int_{\boldsymbol{R}} \partial_y^3 W \partial_y^4 W \partial_y^6 W \, dy - \underbrace{21\beta_\tau \int_{\boldsymbol{R}} \partial_y^2 W \partial^5 W \partial_y^6 W \, dy}_{II} \\ &- \underbrace{\beta_\tau e^{(3\alpha-1)s} \int_{\boldsymbol{R}} \partial_y^6 W (-\Delta)^\alpha \partial_y^6 W \, dy}_{III}.\end{aligned}$$

The operator $(-\Delta)^\alpha$ is self-adjoint on L^2, allowing us to drop III from subsequent estimates. Integration by parts and the supposed L^2 decay of $\partial_y^6 W$ gives

$$\begin{aligned} I &= \frac{1}{2}\int_{\boldsymbol{R}} g_W \frac{\partial}{\partial y}(\partial_y^6 W)^2 \, dy \\ &= \frac{\beta_\tau}{2}\left(\int_{\boldsymbol{R}} W \frac{\partial}{\partial y}(\partial_y^6 W)^2 \, dy + e^{s/2}(\kappa - \dot{\xi}) \int_{\boldsymbol{R}} \frac{\partial}{\partial y}(\partial_y^6 W)^2 \, dy\right) + \frac{3}{4}\int_{\boldsymbol{R}} y \frac{\partial}{\partial y}(\partial_y^6 W)^2 \, dy \\ &= -\frac{\beta_\tau}{2}\int_{\boldsymbol{R}} \partial_y W (\partial_y^6 W) \, dy - \frac{3}{4}\|\partial_y^6 W\|_{L^2}^2. \end{aligned}$$

Similarly, we form a total derivative and integrate by parts to obtain

$$II = \frac{21}{2}\beta_\tau \int_{\boldsymbol{R}} \partial_y^3 W (\partial_y^5 W)^2 \, dy.$$

Collecting like terms and using the bootstrap assumption (2.4) to bound the third derivative in L^∞, we obtain the energy equality

$$\begin{aligned}\frac{d}{ds}\|\partial_y^6 W\|_{L^2}^2 &= -\frac{31}{2}\|\partial_y^6 W\|_{L^2}^2 - 13\beta_\tau \int_{\boldsymbol{R}} \partial_y W (\partial_y^6 W)^2\, dy \\ &\quad - 70\beta_\tau \int_{\boldsymbol{R}} \partial_y^3 W \partial_y^5 W \partial_y^6 W\, dy + 21\beta_\tau \int_{\boldsymbol{R}} \partial_y^3 W (\partial_y^5 W)^2\, dy - 2\|\partial_y^6 W\|_{H^{\alpha/2}} \\ &\leqslant \left(-\frac{31}{2} + 13\beta_\tau\right)\|\partial_y^6 W\|_{L^2}^2 + 35\beta_\tau M \|\partial_y^4 W\|_{L^2}\|\partial_y^6 W\|_{L^2} + 21\beta_\tau M\|\partial_y^5 W\|_{L^2}^2.\end{aligned} \tag{2.14}$$

We use Galiardo-Nirenberg to interpolate between $\partial_y^6 W$ in L^2 and the uniform L^2 bound of $\partial_y W$ proven in Lemma 2.3.0.1:

$$\|\partial_y^4 W\|_{L^2} \lesssim \|\partial_y W\|_{L^2}^{2/5}\|\partial_y^6 W\|_{L^2}^{3/5} \lesssim \|\partial_y^6 W\|_{L^2}^{3/5},$$

$$\|\partial_y^5 W\|_{L^2} \lesssim \|\partial_y W\|_{L^2}^{1/5}\|\partial_y^6 W\|_{L^2}^{4/5} \lesssim \|\partial_y^6 W\|_{L^2}^{4/5}.$$

We insert the interpolants into our energy estimate (2.14), take β_τ close to 1 via (2.8), and apply Young's inequality to find

$$\frac{d}{ds}\|\partial_y^6 W\|_{L^2}^2 \lesssim \left(-\frac{31}{2} + 13\beta_\tau\right)\|\partial_y^6 W\|_{L^2}^2 + M\beta_\tau \|\partial_y^6 W\|_{L^2}^{8/5} \lesssim M - 2\|\partial_y^6 W\|_{L^2}^2.$$

Integrating this differential inequality gives

$$\|\partial_y^6 W\|_{L^2}^2 \lesssim \left(1 - e^{-2c(s-s_0)}\right) M + \|\partial_y^6 W^0\|_{L^2}^2 e^{-2c(s-s_0)}. \tag{2.15}$$

Using our initial data assumption (2.15) and taking M sufficiently large, we obtain

$$\|\partial_y^6 W\|_{L^2}^2 \lesssim M \leqslant M^{3/2}. \tag{2.16}$$

Finally, by Gagliardo-Nirenberg

$$\|\partial_y^3 W\|_{L^\infty} \lesssim \|\partial_y W\|_{L^\infty}^{5/9}\|\partial_y^6 W\|_{L^2}^{4/9} \lesssim M^{1/3}. \tag{2.17}$$

This closes the bootstrap (2.4) upon choosing M sufficiently large.

L^∞ Interpolation of the Intermediary Derivatives. We avoid bootstrapping the global bounds on $\partial_y^2 W$ and $\partial_y^3 W$ by interpolation. Gagliardo-Nirenberg gives the following bounds

$$\|\partial_y^5 W\|_{L^\infty} \lesssim \|\partial_y W\|_{L^\infty}^{1/9}\|\partial_y^5 W\|_{L^2}^{8/9} \lesssim M^{\frac{2}{3}} \tag{2.18a}$$

$$\|\partial_y^4 W\|_{L^\infty} \lesssim \|\partial_y W\|_{L^\infty}^{1/3}\|\partial_y^5 W\|_{L^2}^{2/3} \lesssim M^{\frac{1}{2}} \tag{2.18b}$$

$$\|\partial_y^2 W\|_{L^\infty} \lesssim \|\partial_y W\|_{L^\infty}^{7/9}\|\partial_y^5 W\|_{L^2}^{2/9} \lesssim M^{\frac{1}{6}} \tag{2.18c}$$

With control on $\partial_y^2 W$ and $\partial_y^3 W$ in L^∞ from (2.18c) and (2.17), we can combine the identity (2.7b) and interpolation inequality (2.6) for $(-\Delta)^\alpha$ to obtain

$$e^{s/2}|\kappa - \dot{\xi}| \lesssim \frac{e^{(3\alpha-1)s}}{5} \|\partial_x^2 W\|_{L^\infty}^{1-2\alpha} \|\partial_x^3 W\|_{L^\infty}^{2\alpha} \leqslant e^{\frac{9}{10}(3\alpha-1)s}, \tag{2.19}$$

sacrificing powers of ε to absorb the constant.

2.3.4. Lagrangian Trajectories. We prove upper and lower bounds on the Lagrangian trajectories of (2.4) which allow us to convert spatial decay into temporal decay.

The flowmap evolution of (2.4) is

$$\frac{d}{ds}\Phi^{y_0} = \frac{3}{2}\Phi^{y_0} + \beta_\tau \left(W^{y_0} + e^{\frac{s}{2}}(\kappa - \dot{\xi}) \right), \tag{2.20}$$

where Φ^{y_0} is the trajectory emanating from the label y_0 and $W^{y_0}(s) := W(\Phi^{y_0}(s), s)$.

Lemma 2.3.0.2. ***(Upper Bound on Trajectories).*** *Let $x_0 \in \boldsymbol{R}$, then trajectories are bounded above by*

$$|\Phi^{x_0}(s)| \leqslant \left(|x_0| + \frac{3}{2} C \varepsilon^{-1/2} \right) e^{\frac{3}{2}(s-s_0)}, \tag{2.21}$$

with C a positive constant depending linearly on M.

Proof. The flowmap equation (2.20) implies that

$$\frac{d}{ds}\left[\Phi^{x_0} e^{-3s/2} \right] = \beta_\tau e^{-3s/2} \left(W^{x_0} + e^{s/2}(\kappa - \dot{\xi}) \right).$$

We integrate in time, estimating β_τ with (2.8), and apply (2.10), (2.19) to obtain

$$\begin{aligned} |\Phi^{x_0}(s)| &\leqslant |x_0| e^{\frac{3}{2}(s-s_0)} + e^{3s/2} \int_{s_0}^{s} e^{-\frac{3}{2}s'} \beta_\tau \left(W^{x_0}(s') + e^{s'/2}(\kappa - \dot{\xi}) \right) ds' \\ &\leqslant \varepsilon |x_0| e^{\frac{3}{2}s} + \frac{3}{2}\varepsilon^{\frac{3}{2}s} \int_{s_0}^{s} e^{-s'} \left(\|u_0\|_{L^\infty} + 6M \right) + e^{-\frac{3}{2}s'} \varepsilon^{\frac{9}{10}(1-3\alpha)} \, ds', \end{aligned}$$

from which the result follows. ■

Next we prove two lower bounds on the Lagrangian trajectories. The first bound serves only to guarantee that all trajectories emanating from outside the Taylor region will eventually take off with our optimal bound (2.23).

Lemma 2.3.0.3. *Let $h \leqslant |x_0| < \infty$. The trajectories are bounded below by*

$$|\Phi^{x_0}(s)| \geqslant |x_0| e^{\frac{1}{5}(s-s_0)}. \tag{2.22}$$

Proof. The proof is contained in [**112**], replacing the author's bound for $e^{\frac{s}{2}}(\kappa - \dot{\xi})$ by our bound (2.19) and taking ε sufficiently small. ∎

To close the bootstraps, this $1/5$ bound proves insufficient; we require a stronger lower bound to prove Lemma 2.4.0.1 and to close the bootstrap (2.1b). In particular we must have that the trajectories eventually take off faster than $e^{\frac{3}{5}s}$. The following estimate is optimal in light of Lemma 2.3.0.2 and the transformation (2.2).

Lemma 2.3.0.4. *Let $1 \leqslant |x_0| < \infty$. The trajectories are bounded below by*

$$|\Phi^{x_0}(s)| \geqslant \left(|x_0|^{2/3} - \frac{2C}{3} \right) e^{\frac{3}{2}(s-s_0)} > 0. \tag{2.23}$$

The constant C is

$$C = (1 + 2\varepsilon^{\frac{1}{2}(1-3\alpha)}) \left(1 + \frac{1}{h^{1/3}} \left(\varepsilon^q \langle h \rangle + \varepsilon^{\frac{1}{2}(1-3\alpha)} \right) \right).$$

Proof. Multiply the flowmap (2.20) by Φ^{x_0} to obtain

$$\frac{d}{ds} |\Phi^{x_0}|^2 = 3|\Phi^{x_0}|^2 + 2\beta_\tau \Phi \left(W + e^{\frac{s}{2}}(\kappa - \dot{\xi}) \right).$$

Recall that Ψ is the self-similar solution to Burgers equation (2.7) and observe that in light of (2.13) and (2.1b) we have

$$|W(x,s)| \leqslant |\Psi(x)| + |\widetilde{W}(x,s)| \leqslant |x|^{1/3} + \varepsilon^q \langle x \rangle^{1/3} \leqslant \left(1 + \varepsilon^q \frac{\langle h \rangle^{1/3}}{h^{1/3}} \right) |x|^{1/3}.$$

In a similar vein, (2.19) implies that

$$|e^{\frac{s}{2}}(\kappa - \dot{\xi})| \leqslant \frac{\varepsilon^{\frac{1}{2}(1-3\alpha)}}{h^{1/3}} |x^{1/3}|.$$

Estimating β_τ as in (2.8) and taking C as in the statement of the lemma, we obtain the differential inequality

$$\frac{d}{ds} |\Phi^{x_0}|^2 - 3|\Phi^{x_0}|^2 \geqslant -2C |\Phi^{x_0}|^{4/3}.$$

This inequality is separable in the variable $|\Phi^{x_0}|^2 e^{-3s}$, and integrating from s_*, $s_0 \leqslant s_* < s$ yields (2.23) upon taking ε sufficiently small. ∎

Together, these two lemmas show that all trajectories with starting labels $h \leqslant |x_0|$ will eventually take off like $e^{\frac{3}{2}s}$, i.e. $|\Phi^{x_0}(s)| \lesssim e^{\frac{3}{2}s}$ whenever $h \leqslant |x_0|$.

2.4. Bootstraps: Closing The L^∞ Estimates

2.4.1. Modulation Variables. Our interpolation inequality (2.6) for $(-\Delta)^\alpha$ and the estimates (2.7), (2.18c) give

$$|\dot\tau| \leqslant e^{(3\alpha-1)s}\|\partial_y W\|_{L^\infty}^{1-2\alpha}\|\partial_y^2 W\|_{L^\infty}^{2\alpha} \lesssim M^{\frac{1}{3}\alpha}e^{(3\alpha-1)s} \leqslant \frac{1}{2}e^{\frac{8}{9}(3\alpha-1)s} \tag{2.1}$$

upon choosing ε sufficiently small. We apply the fundamental Theorem of calculus and recall that $\tau(-\varepsilon)=0$ to find

$$|\tau(t)| \leqslant \int_{-\varepsilon}^t |\dot\tau(t')|\,dt' \leqslant \frac{1}{2}\varepsilon^{\frac{8}{9}(1-3\alpha)}(T_*+\varepsilon) \leqslant \frac{1}{2}\varepsilon^{1-3\alpha}+\frac{1}{2}\varepsilon^{1+\frac{1}{2}(1-3\alpha)} \leqslant \frac{3}{4}\varepsilon^{\frac{8}{9}(1-3\alpha)}.$$

This closes the τ bootstraps (2.6a) and (2.6b).

The ξ bootstrap is equally simple. By the identity (2.7b),

$$\begin{aligned}|\dot\xi| &\leqslant |\kappa| + \frac{e^{s(3\alpha-\frac{3}{2})}}{|\partial_y^3 W(0,s)|}((-\Delta)^\alpha\partial_y^2 W)(0,s)\\ &\leqslant M + \frac{1}{5}e^{(3\alpha-\frac{3}{2})s}M^{\frac{1}{6}+\frac{\alpha}{3}}\\ &\leqslant 2M,\end{aligned} \tag{2.2}$$

where we have chosen ε sufficiently small and M sufficiently large. Integrating (2.2) in time and using the fundamental theorem of calculus, recalling that $|\xi(-\varepsilon)|=0$,

$$|\xi(t)| \leqslant \int_{-\varepsilon}^t |\dot\xi|\,dt' \leqslant 2M(T_*+\varepsilon) \leqslant 2M(\varepsilon^{1+2\alpha}+\varepsilon) \leqslant 4M\varepsilon.$$

This closes the two ξ bootstraps (2.6a) and (2.6b).

2.4.2. Taylor Region.

(Fourth Derivative on $0 \leqslant |y| \leqslant h$). The fourth derivative of $\widetilde{W}$ evolves according to (2.28), which we recall here

$$\underbrace{\left(\partial_s + \frac{11}{2} + 5\beta_\tau\partial_y W\right)}_{\mathcal{D}(y)}\partial_y^4\widetilde{W} + g_W\partial_y^4\widetilde{W} = -\beta_\tau\mathcal{F}.$$

$\mathcal{D}$ is the damping and $\mathcal{F}$ is the forcing, which is given explicitly in (2.4).

Using (2.7) and (2.8) we bound the damping below by

$$\mathcal{D} \geqslant \frac{11}{2} - 5\beta_\tau \geqslant \frac{1}{4}. \tag{2.3}$$

The forcing is given by

$$
\begin{aligned}
\mathcal{F} := -\underbrace{\beta_\tau e^{(3\alpha-1)s}(-\Delta)^\alpha \partial_y^4 W}_{I} + \underbrace{\beta_\tau \left(\widetilde{W}\Psi^{(5)} + 4\partial_y \widetilde{W}\Psi^{(4)} + 8\partial_y^2\widetilde{W}\Psi^{(3)} + 10\partial_y^3\widetilde{W}\Psi'' + 11\partial_y^2\widetilde{W}\partial_y^3\widetilde{W}\right)}_{II} \\
+ \underbrace{\beta_\tau e^{s/2}(\kappa - \dot{\xi})\partial_y^5\Psi}_{III} + \underbrace{\beta_\tau \dot{\tau}\left(\Psi\partial_y^5\Psi + 5\Psi'\Psi^{(4)} + 10\Psi''\Psi^{(3)}\right)}_{IV}
\end{aligned}
\tag{2.4}
$$

Term I is estimated by combining our interpolation estimate (2.6) on the fractional Laplacian with Gagliardo-Nirenberg

$$
|I| \lesssim \beta_\tau e^{(3\alpha-1)s}\|\partial_y W\|_{L^\infty}^{\frac{1}{3}-\frac{4}{9}\alpha}\|\partial_y^6 W\|_{L^2}^{\frac{2}{3}+\frac{4}{9}\alpha} \leqslant \beta_\tau e^{(3\alpha-1)s}M^{\frac{1}{2}+\frac{\alpha}{3}} \leqslant \varepsilon^{\frac{8}{9}(1-3\alpha)},
$$

where in the second inequality we have used (2.7) and (2.16) and in the final inequality we have sacrificed a power of ε. Recalling that $h = (\log M)^{-2}$, the term II can be bounded by the bootstrap assumptions (2.1)-(2.3) in the Taylor region

$$
\begin{aligned}
|II| &\lesssim \varepsilon^{\frac{8}{9}(1-3\alpha)} + \log M \varepsilon^{\frac{7}{9}(1-3\alpha)}(h^4 + h^3 + h^2 + h) \\
&\lesssim \varepsilon^{\frac{8}{9}(1-3\alpha)} + (\log M)^{-1}\varepsilon^{\frac{7}{9}(1-3\alpha)}.
\end{aligned}
$$

The estimate (2.19) along with the explicit property (2.14) on $\Psi^{(5)}$ imply that

$$
|III| = \left| e^{s/2}(\kappa - \dot{\xi})\Psi^{(5)}\right| \lesssim \varepsilon^{\frac{9}{10}(1-3\alpha)} \leqslant \varepsilon^{\frac{8}{9}(1-3\alpha)}.
$$

The term IV is quadratic in Ψ so (2.6) and (2.14) give

$$
|IV| \lesssim \varepsilon^{\frac{1}{2}(1-3\alpha)} \leqslant \varepsilon^{\frac{8}{9}(1-3\alpha)}.
$$

In light of (2.3) and our forcing estimates, we write

$$
\partial_s \partial_y^4 \widetilde{W} + \frac{1}{4}\partial_y^4\widetilde{W} + g_W \partial_y^4 \widetilde{W} \geqslant -\beta_\tau \mathcal{F}.
$$

Composing this inequality with the flowmap Φ^{y_0} and integrating gives us

$$
\begin{aligned}
|\partial_y^4 \widetilde{W}^{y_0}(s)| &\leqslant |\partial_y^4\widetilde{W}^{y_0}(s)|e^{-\frac{s}{4}} + e^{-\frac{s}{4}}\int_{s_0}^{s} e^{\frac{s}{4}}\mathcal{F}^{y_0}\,ds' \\
&\leqslant \varepsilon^{1-3\alpha} + C(\varepsilon^{\frac{8}{9}(1-3\alpha)} + (\log M)^{-1}\varepsilon^{\frac{7}{9}(1-3\alpha)}) \\
&\leqslant \frac{1}{2}\varepsilon^{\frac{7}{9}},
\end{aligned}
\tag{2.5}
$$

where we have used the Lagrangian trajectory lower bound (2.23) and the initial condition (2.14). This closes the bootstrap (2.3c) for $\partial_y^4\widetilde{W}$.

(Third Derivative at 0). The third derivative evolves according to equation (2.24), which reads

$$\left(\partial_s + 4 + 4\beta_\tau \partial_y W\right)\partial_y^3 W + g_W \partial_y^4 W = -\beta_\tau \left[e^{(3\alpha-1)s}(-\Delta)^\alpha \partial_y^3 W + 3(\partial_y^2 W)^2\right].$$

Setting $y = 0$ and using the constraints (2.6) we obtain the identity

$$\partial_s \partial_y^3 W^0 = \underbrace{4\dot{\tau}\beta_\tau \partial_y^3 W^0}_{I} - \underbrace{\beta_\tau e^{s/2}(\kappa - \dot{\xi})\partial_y^4 W^0}_{II} - \underbrace{\beta_\tau e^{(3\alpha-1)s}(-\Delta)^\alpha [\partial_y^3 W]^0}_{III}.$$

Using our β_τ estimate (2.8), the fact that $|\partial_y^3 W^0(s)| \leqslant 7$, and the $\dot{\tau}$ bootstrap closure (2.1), we find

$$|I| \leqslant 28 e^{\frac{8}{9}(3\alpha-1)s}.$$

Next we apply the estimates (2.19) and (2.8), as well as our L^∞ control (2.18b) on $\partial_y^4 W$ to find

$$|II| \lesssim 2M^{\frac{1}{2}} e^{\frac{9}{10}(3\alpha-1)s} \leqslant e^{\frac{8}{9}(3\alpha-1)s}$$

upon taking ε sufficiently small. Finally, we estimate

$$|III| \lesssim 2M^{\frac{1}{3}(1+\alpha)}(1-\alpha)\, e^{(3\alpha-1)s} \leqslant e^{\frac{8}{9}(3\alpha-1)s}$$

by applying our interpolation estimate (2.6) for the fractional Laplacian, (2.8), our L^∞ bounds (2.17), (2.18b), and taking ε sufficiently small.

We record the following fact

$$|\partial_s \partial_y^3 W(0,s)| \leqslant 3e^{\frac{8}{9}(3\alpha-1)s} \leqslant 3\varepsilon^{\frac{8}{9}(1-3\alpha)}, \tag{2.6}$$

which we will need to use later in our proof.

Recalling that $\partial_y^3 W(0, s_0) = 6$ and applying the Fundamental theorem of calculus to the estimate above, we see that

$$\begin{aligned} |\partial_y^3 W(0,s) - 6| &\leqslant \int_{-\varepsilon}^{t} 3\varepsilon^{\frac{8}{9}(1-3\alpha)}\, dt' \leqslant 3\varepsilon^{\frac{8}{9}(1-3\alpha)}(T_* + \varepsilon) \leqslant 3\varepsilon^{(\frac{8}{9}+\frac{8}{9})(1-3\alpha)} + 3\varepsilon^{(1+\frac{8}{9})(1-3\alpha)} \\ &\leqslant 6\varepsilon^{\frac{16}{9}(1-3\alpha)}. \end{aligned} \tag{2.7}$$

Taking ε small closes the bootstrap (2.5).

(Zeroth, First, Second, and Third Derivative on $0 \leqslant |y| \leqslant h$). By the fundamental theorem of calculus, our bootstrap (2.5), and the estimate (2.5),

$$
\begin{aligned}
|\partial_y^3 \widetilde{W}| &= \left| \partial_y^3 \widetilde{W}(0,s) + \int_0^y \partial_y^4 \widetilde{W}(y',s)\, dy' \right| \\
&\leqslant 6\varepsilon^{\frac{9}{10}(1-3\alpha)} + \frac{1}{2}\varepsilon^{\frac{7}{9}(1-3\alpha)} h \\
&\leqslant \frac{1}{2}\left(\varepsilon^{\frac{8}{9}(1-3\alpha)} + \log M \varepsilon^{\frac{7}{9}(1-3\alpha)} \right) h,
\end{aligned}
$$

which closes (2.3b) on $\partial_y^3 \widetilde{W}$. Recall that our constraints (2.6) at the origin are

$$
\partial_y^j \widetilde{W}(0,s) = 0
$$

for $j = 0, 1, 2$. We can iterative apply the fundamental theorem of calculus to estimate

$$
|\partial_y^j \widetilde{W}| = \left| \partial_y^j \widetilde{W}(0,s) + \int_0^y \partial_y^{j+1} \widetilde{W}(y',s)\, dy' \right| \leqslant \frac{1}{2}\left(\varepsilon^{\frac{8}{9}(1-3\alpha)} + \log M \varepsilon^{\frac{7}{9}(1-3\alpha)} \right) h^{4-j},
$$

which closes the bootstraps (2.1a), (2.2a), and (2.3a).

2.4.3. Outside the Taylor Region.

To close the bootstraps on the rest of $\boldsymbol{R}$ we will utilize the trajectory framework presented in Section 2.1.4.1.

(Zeroth Derivative on $h \leqslant |y| < \infty$). We consider the weighted variable $V := \langle x \rangle^{-1/3} \widetilde{W}$. The evolution equation for V is obtained from (2.26) after a short computation

$$
\Big(\partial_s \underbrace{- \frac{1}{2} + \beta_\tau \Psi' + \frac{g_W}{3} y \langle y \rangle^{-2}}_{\mathcal{D}(y)} \Big) V + g_W V' = -\beta_\tau \langle y \rangle^{-\frac{1}{3}} F_{\widetilde{W}}, \tag{2.8}
$$

where the forcing is given below in (2.10). Closing the bootstrap (2.1b) now amounts to showing that $|V(x,s)| < \varepsilon^q$ uniformly in self similar time, for $h \leqslant |x| < \infty$. From Section 2.1.4.1, if we can control the damping from below and the forcing from above on our region $h \leqslant |x| < \infty$, then we can establish global (in time) estimates by carefully following the trajectories through our region.

We begin by bounding our damping below by (2.8), (2.14), (2.19) and our bootstrap assumption (2.1b),

$$
\begin{aligned}
\mathcal{D}(y) &\geqslant \frac{1}{2} - (1 + 2\varepsilon^{\frac{1}{2}(1-3\alpha)}) \langle y \rangle^{-2/3} - \frac{1}{3}\left(\beta_\tau W + \beta_\tau e^{s/2}(\kappa - \dot{\xi}) + \frac{3}{2} y \right) y \langle y \rangle^{-2} \\
&\geqslant \frac{1}{2}\langle y \rangle^{-2} - (1 + 2\varepsilon^{\frac{1}{2}(1-3\alpha)}) \langle y \rangle^{-2/3} - \frac{2}{3}(1 + \varepsilon^q) y \langle y \rangle^{-5/3} - \varepsilon^{\frac{1}{2}(1-3\alpha)} |y| \langle y \rangle^{-2} \\
&\geqslant -(2 + 3\varepsilon^{\frac{1}{2}(1-3\alpha)} + \frac{2}{3}\varepsilon^{\frac{2}{3}(1-3\alpha)}) \langle y \rangle^{-2/3} \\
&\geqslant -3\langle y \rangle^{-2/3}.
\end{aligned}
$$

This inequality holds upon taking ε sufficiently small.

Let s_* be the first time that the Trajectory Φ^{x_0} enters the region $h \leqslant |y| < \infty$. By composing the damping with the lower bound of the trajectories and integrating in time we find that

$$\int_{s_*}^{s} \langle \Phi^{y_0}(s') \rangle^{-\frac{2}{3}} \, ds' \leqslant \int_{s_*}^{\infty} \langle |y_0| e^{\frac{2}{5}(s-s_*)} \rangle^{-2/3} \, ds' \leqslant \frac{13}{2} \log\left(\frac{1}{h}\right), \tag{2.9}$$

which is the same estimate obtained in [15]. Using (2.9), we set

$$\lambda_{\mathcal{D}} := e^{-\int_{s_0}^{s} \mathcal{D}(\Phi^{y_0}(s')\, ds'} \leqslant e^{\frac{39}{2}\log\left(\frac{1}{h}\right)} = h^{-\frac{39}{2}}.$$

Next we bound the forcing above in L^∞. Recall that the forcing for equation (2.8) is given by (2.26)

$$F_{\widetilde{W}} = \underbrace{e^{-\frac{s}{2}}\dot{\kappa}}_{I} + \underbrace{e^{(3\alpha-1)s}(-\Delta)^\alpha W}_{II} + \underbrace{\Psi'(\dot{\tau}\Psi)}_{III} + \underbrace{\Psi' e^{\frac{s}{2}}(\kappa - \dot{\xi})}_{IV}. \tag{2.10}$$

We also note that $\langle y \rangle^{-1/3} \circ \Phi^{y_0}(s) \leqslant e^{-\frac{1}{2}(s-s_0)}$ as a consequence of our lower bound (2.23) on the trajectories.

We estimate term by term. For the first term we use identity (2.7b), the interpolation inequality (2.6), (2.8), (2.10), and take ε sufficiently small

$$\begin{aligned}
\beta_\tau |\langle \Phi^{y_0}(s) \rangle^{-1/3} I| &\lesssim \beta_\tau \langle e^{\frac{3}{2}(s-s_*)} \rangle^{-1/3} e^{(3\alpha-1)s} \left[(-\Delta)^\alpha W^0 - \frac{(-\Delta)^\alpha \partial_y^2 W^0}{\partial_y^3 W^0} \right] \\
&\lesssim \beta_\tau \varepsilon^{\frac{1}{2}} e^{(3\alpha-\frac{3}{2})s} \left(\|W\|_{L^\infty}^{1-2\alpha} \|\partial_y W\|_{L^\infty}^{2\alpha} + \frac{1}{5} \|\partial_y^2 W\|_{L^\infty}^{1-2\alpha} \|\partial_y^3 W\|_{L^\infty}^{2\alpha} \right) \\
&\lesssim \varepsilon^{\frac{1}{2}} e^{(3\alpha-\frac{3}{2})s} \left(e^{\frac{s}{2}-\alpha s} + \frac{1}{5} M^{\frac{1}{6}-\frac{\alpha}{3}} \right) \\
&\leqslant \frac{1}{4} e^{(2\alpha-1)s}.
\end{aligned}$$

Note that the $e^{s/5}$ lower bound (2.22) on the trajectories is insufficient to close the above estimate.

For the next term we use our interpolation estimate (2.6), (2.10), (2.8), and sacrifice powers of ε to get

$$\beta_\tau |\langle \Phi^{y_0}(s) \rangle^{-1/3} II| \lesssim \varepsilon^{1/2} e^{(3\alpha-\frac{3}{2})s} \|W\|_{L^\infty}^{1-2\alpha} \|\partial_y W\|_{L^\infty}^{2\alpha} \leqslant \frac{1}{4} e^{(2\alpha-1)s}.$$

We estimate the third term by using (2.14) twice, (2.8), our bootstrap assumption (2.6) on $\dot{\tau}$, and taking ε small

$$\beta_\tau |\langle \Phi^{y_0}(s) \rangle^{-1/3} III| \lesssim \dot{\tau} \varepsilon^{1/2} e^{-\frac{s}{2}} \langle \Phi^{y_0}(s) \rangle^{-\frac{1}{3}} \leqslant \frac{1}{4} e^{-s}.$$

For the final term we use (2.19), (2.14), and take ε small to find

$$\beta_\tau |\langle \Phi^{y_0}(s) \rangle^{-1/3} IV| \leqslant 2\varepsilon^{1/2} \frac{e^{(3\alpha-\frac{3}{2})s}}{5} \|\partial_y^2 W\|_{L^\infty}^{1-2\alpha} \|\partial_y^3 W\|_{L^\infty}^{2\alpha} \langle \Phi^{y_0}(s) \rangle^{-2/3} \leqslant \frac{1}{4} e^{(3\alpha-\frac{5}{2})s}.$$

Together these four estimates give

$$|\beta_\tau \langle y\rangle^{-1/3} F_{\widetilde{W}}| \leqslant e^{(2\alpha-1)s}. \tag{2.11}$$

Since $\alpha < {}^1\!/\!_3$, we have temporal decay in our forcing terms. Inserting the damping estimate (2.9) and the forcing estimate (2.11) into (2.5) gives

$$\begin{aligned} |V^{y_0}(s)| &\lesssim \lambda_D |V(y_0)| + \lambda_D \int_{s_*}^{s} e^{-(1-2\alpha)s'}\, ds' \\ &\lesssim \lambda_D \left(|V(y_0)| + e^{(2\alpha-1)s_*} \right) \end{aligned}$$

As in the discussion in Section 2.1.4.1, we have two cases; either $s_* = s_0$, and $h \leqslant y_0 < \infty$, or $s_* > s_0$ and $y_0 = h$. In the first case we use our initial data assumption (2.12) to find that

$$|V^{y_0}(s)| \lesssim \lambda_D \left(\varepsilon^{1-3\alpha} \langle y_0\rangle^0 + \varepsilon^{-(1-2\alpha)s_0} \right) \leqslant \varepsilon^{\frac{1}{2}(1-2\alpha)}.$$

In the second case we apply our bootstrap (2.1a) for the region $0 \leqslant |y| \leqslant h$ and estimate

$$|V^{y_0}(s)| \lesssim \lambda_D \left(\left(\varepsilon^{\frac{8}{9}(1-3\alpha)} + \log M \varepsilon^{\frac{7}{9}(1-3\alpha)} \right) h^4 + \varepsilon^{-(1-2\alpha)s_*} \right).$$

In both cases we may take ε sufficiently small to get

$$|V^{y_0}(s)| \leqslant \frac{1}{2}\varepsilon^q.$$

This closes the zeroth derivative bootstrap (2.1) over the region $h \leqslant |y| < \infty$.

(First Derivative on $h \leqslant |y| \leqslant e^{ms}$). We use the same strategy used to close the zeroth derivative bootstrap. Making the weighted change of variables $V := \langle y\rangle^{2/3}\partial_y \widetilde{W}$, a short computation using (2.27) yields the following evolution equation for V

$$\partial_s V + \Big(\underbrace{1 + \beta_\tau(\partial_y \widetilde{W} + 2\Psi') - \frac{2}{3} y \langle y\rangle^{-2} g_W}_{\mathcal{D}(y)} \Big) V + g_W \partial_y V = -\beta_\tau \langle y\rangle^{2/3} F_{\partial_y \widetilde{W}} \tag{2.12}$$

where $F_{\widetilde{W}'}$ is the forcing term from (2.27) which is given below in (2.14). Closing the bootstrap (2.2b) now amounts to showing that $|V(x,s)| < \varepsilon^{\frac{3}{4}\ell}$ uniformly in s for all $h \leqslant |x| \leqslant e^{ms}$.

We begin by bounding the damping below. Using (2.8), (2.14), the bootstraps (2.1) and (2.2), and (2.19) we find

$$\begin{aligned}\mathcal{D}(y) &= -1 - \beta_\tau(\partial_y \widetilde{W} + 2\Psi') + \frac{2}{3}y\langle y\rangle^{-2}\left(\frac{3}{2}y + \beta_\tau\left(W + e^{s/2}(\kappa - \dot{\xi})\right)\right)\\ &\geqslant -\langle y\rangle^{-2} - (1 + 2\varepsilon^{\frac{1}{2}(1-3\alpha)})\left(\varepsilon^{\frac{3}{4}\ell}\langle y\rangle^{-2/3} + 2\langle y\rangle^{-2/3}\right.\\ &\qquad\qquad \left.+\frac{2}{3}|y|\langle y\rangle^{-5/3} + \varepsilon^{\frac{1}{2}(1-3\alpha)}\frac{2}{3}(1+)|y|\langle y\rangle^{-2}\right)\\ &\geqslant -\langle y\rangle^{-2} - 5\langle y\rangle^{-2/3}\\ &\geqslant -6\langle y\rangle^{-2/3}.\end{aligned}$$

Our estimate (2.9) still applies to the current damping lower bound for the first derivative. Therefore we have

$$\lambda_{\mathcal{D}} := e^{-\int_{s_0}^{s}\mathcal{D}(\Phi^{y_0}(s'))\,ds'} \leqslant e^{39\log(\frac{1}{h})} = h^{-39}. \tag{2.13}$$

The forcing for V in this case is given by

$$F_{\partial_y\widetilde{W}} := \underbrace{e^{(3\alpha-1)s}(-\Delta)^{\alpha}\partial_y W}_{I} + \underbrace{e^{s/2}(\kappa - \dot{\xi})\Psi''}_{II} + \underbrace{\widetilde{W}\Psi''}_{III} + \underbrace{\dot{\tau}\Psi\Psi''}_{IV} + \underbrace{(\Psi')^2}_{V} \tag{2.14}$$

We bound the first term using (2.8), the interpolation estimate (2.6), and by taking ε sufficiently small.

$$\begin{aligned}\beta_\tau|\langle y\rangle^{2/3} I| &\lesssim e^{\frac{2}{3}ms}e^{(3\alpha-1)s}\|\partial_y W\|_{L^\infty}^{1-2\alpha}\|\partial_y^2 W\|_{L^\infty}^{2\alpha}\\ &\lesssim M^{\frac{1}{6}}e^{(\frac{2}{3}m+3\alpha-1)s}.\end{aligned}$$

We choose m such that the exponent in the exponential above remains positive for all $0 < \alpha < 1/3$. Remember that we are aiming to prove that V is bounded by some multiple of $\varepsilon^{2/3(1-3\alpha)}$, which means we must have $m < \frac{3}{8}(1-3\alpha)$. This informs the choice of m in (2.8).

For the second term, we use identity (2.7b), (2.14), the interpolation estimate (2.6), the L^∞ control (2.17),(2.18b), and then take ε small to find

$$\begin{aligned}\beta_\tau|\langle y\rangle^{2/3} II| &\leqslant \beta_\tau\frac{e^{(3\alpha-1)s}}{\partial_y^3 W(0,s)}|(-\Delta)^{\alpha}\partial_y^2 W(0,s)|\langle y\rangle^{-1}\\ &\lesssim M^{\frac{1}{6}(1+\alpha)}e^{(3\alpha-1-\frac{3}{2})s}.\end{aligned}$$

We apply the bootstrap (2.1a) for $\widetilde{W}$, use (2.14), and estimate the trajectories with (2.22) to get

$$\begin{aligned}\beta_\tau\left|\left(\langle y\rangle^{2/3} III\right)\circ\Phi^{y_0}(s)\right| &\leqslant \beta_\tau\varepsilon^{\frac{1}{2}q}\langle\Phi^{y_0}(s)\rangle^{-2/3}\\ &\lesssim \varepsilon^{\frac{1}{2}q}e^{-s}.\end{aligned}$$

Next, (2.14), (2.8), and taking ε small yields

$$\beta_\tau \left| \left(\langle y \rangle^{2/3} IV \right) \circ \Phi^{y_0}(s) \right| \leqslant \beta_\tau \dot{\tau} \langle \Phi^{y_0}(s) \rangle^{-4/3} \lesssim \varepsilon^{\frac{8}{9}(1-3\alpha)} e^{-2s}.$$

Finally, (2.14), (2.8), and (2.6b) imply that

$$\beta_\tau \left| \left(\langle y \rangle^{2/3} V \right) \circ \Phi^{y_0}(s) \right| \leqslant \beta_\tau \langle \Phi^{y_0}(s) \rangle^{-2/3} \lesssim e^{-s}.$$

To summarize, we have obtained the following forcing bound

$$|F_{\partial_y \widetilde{W}} \circ \Phi^{y_0}(s)| \lesssim e^{-\ell s}, \tag{2.15}$$

with ℓ as in (2.8).

We once again apply the trajectory framework from Section 2.1.4.1 to our equation (2.12) for V. Plugging our damping estimate (2.13) and our forcing estimate (2.15) into (2.5) we obtain

$$\begin{aligned} |V^{y_0}(s)| &\lesssim h^{-39} |V(y_0)| + \lambda_D \int_{s_*}^{s} e^{-\ell s'} \, ds' \\ &\lesssim h^{-39} \left(|V(y_0)| + e^{-\ell s_*} \right) \end{aligned}$$

As in our closure of the zeroth derivative bootstrap and the discussion of our transport framework from Section 2.1.4.1, we have two cases. In the first case, $s_* = s_0$ and $h \leqslant y_0 \leqslant \varepsilon^{-m}$ and we use our initial data assumption (2.13) to find that

$$|V^{y_0}(s)| \lesssim h^{-39} \left(\varepsilon^{\frac{3}{4}\ell} \langle y_0 \rangle^{-2/3} + \varepsilon^{\ell} \right) \lesssim M^{20} \left(\varepsilon^{\frac{3}{4}\ell} + \varepsilon^{\ell} \right) \leqslant \frac{3}{4} \varepsilon^{\frac{3}{4}\ell},$$

upon choosing ε sufficiently small.

In the second case we have that $s_* > s_0 = -\log \varepsilon$, and (2.2) means $V^{y_0}(s_*) = V(h)$. This gives

$$|V^{y_0}(s)| \lesssim M^{20} \left(\varepsilon^{\frac{2}{3}(1-3\alpha)} h^2 \langle h \rangle^{-2} + \varepsilon^{\ell} \right) \leqslant \frac{3}{4} \varepsilon^{\frac{3}{4}\ell}$$

(since $\varepsilon^{\frac{3}{4}\ell} < \varepsilon^{\frac{2}{3}(1-3\alpha)}$ for all α). Thus we have closed the first derivative bootstrap (2.2) on the region $h \leqslant |y| \leqslant e^{ms}$.

(First Derivative on $e^{ms} \leqslant |x| < \infty$). We prove a short fact about the temporal decay of the fractional Laplacian along trajectories.

Lemma 2.4.0.1. *Suppose $e^{ms} \leqslant |y| < \infty$. Then*

$$(-\Delta)^\alpha [\partial_y W](\Phi^y(s)) \lesssim e^{-s}$$

for all s sufficiently large.

Proof. We use the singular integral representation (2.2) for $(-\Delta)^\alpha$. Let $(-\Delta)^\alpha_{[a,b]} u := C_\alpha \int_{a \leqslant |\eta| \leqslant b} \cdots \, d\eta$ and decompose the fractional Laplacian as follows

$$(-\Delta)^\alpha \partial_y W = \underbrace{(-\Delta)^\alpha_{[0,h]} \partial_y \widetilde{W}}_{I} + \underbrace{(-\Delta)^\alpha_{[h,e^{ms}]} \partial_y \widetilde{W}}_{II} + \underbrace{(-\Delta)^\alpha_{[e^{ms},\infty)} \partial_y W}_{III} + \underbrace{(-\Delta)^\alpha_{[0,e^{ms}]} \Psi'}_{IV}.$$

The term IV is majorized by (2.15), that is $|IV| \leqslant \langle y \rangle^{-2/3 - 2\alpha}$. Composing with the lower bound (2.23) on the trajectories shows that $|IV| \lesssim e^{-(1+3\alpha)s}$.

We now go term by term and bound using our bootstraps (2.2). Using (2.2a) for the first term we have

$$\begin{aligned} |I| &\leqslant C_\alpha \int_{0 \leqslant |\eta| \leqslant h} \frac{e^{-s} + C}{|y-\eta|^{1+2\alpha}} \, d\eta \\ &\leqslant 2hC_\alpha (e^{-s} + C) |y-h|^{-1-2\alpha}, \end{aligned}$$

where C is the constant from (2.2a).

Trajectories will either satisfy $\Phi^{y_0}(s) > e^{ms}$ for $|y_0| \geqslant e^{ms_0}$, or s_* is the first time that a trajectory enters $e^{ms} \leqslant y < \infty$, in which case $|y_*| = e^{ms_*}$. In both cases since $e^{ms_0} > 1$ we can apply our trajectory lower bound (2.23). Since $y = \Phi^{y_0}(s) \geqslant |y_0| > e^{ms} > 1 > h$ we have that $|y-h| > |y|$ and composing with trajectories gives $|y-h| \gtrsim e^{\frac{3}{2}s}$. Therefore

$$|I| \lesssim e^{-\frac{3}{2}(1+2\alpha)s}.$$

We remark that this estimate required a lower bound on the trajectories of at least $e^{\frac{3}{5}s}$.

The integrand becomes singular at $y = e^{ms}$. Since $\partial_y W$ and $\partial_y^2 W$ are both L^∞, the interpolation inequality (2.6) guarantees that the integral converges. Because we only care about the asymptotic behavior of these quantities, and since we will be composing with trajectories, we can safely assume that $|x| \geqslant e^{ms} + 1$. The same argument as before applies and trajectories will still take off like $e^{\frac{3}{2}s}$.

To this end we estimate

$$\begin{aligned} |II| &\leqslant C_\alpha \int_{h \leqslant |\eta| \leqslant e^{ms}} \frac{e^{-s} + \varepsilon^{\frac{3}{4}\ell} \langle y \rangle^{-2/3}}{|y-\eta|^{1+2\alpha}} \, d\eta \\ &\leqslant 2C_\alpha (e^{-s} + \varepsilon^{\frac{3}{4}\ell} \langle h \rangle^{-2/3})(e^{ms} - h)|y - e^{ms}|^{1+2\alpha}, \\ |III| &\leqslant 2C_\alpha \int_{e^{ms} \leqslant |\eta| < \infty} \frac{e^{-s}}{|y-\eta|^{1+2\alpha}} \, d\eta \\ &= \frac{2C_\alpha}{\alpha} e^{-s} |y - e^{ms}|^{-2\alpha} \end{aligned}$$

since $|y - e^{ms}| > 1$ our lower bound (2.23) applies (with $y_0 = \Phi(s_*) - e^{ms_*}$) and

$$|II^{y_0}(s)| \lesssim e^{ms} e^{-\frac{3}{2}(1+2\alpha)s}, \qquad |III^{y_0}(s)| \lesssim e^{-(1+3\alpha)s}$$

for s sufficiently large. Collecting the estimates on I, II, and III proves the lemma. ■

We proceed in the same manner as we did when closing the bootstraps for $\widetilde{W}$ on $h \leqslant |y| < \infty$ and $\partial_y \widetilde{W}$ on $h \leqslant |y| \leqslant e^{ms}$. Making the weighted change of variables $V := e^s \partial_y W$, a short computation using (2.22) shows that V satisfies the equation

$$(\partial_s + \beta_\tau \partial_y W)V + g_W \partial_y V = -\beta_\tau e^{3\alpha s}(-\Delta)^\alpha \partial_y W.$$

Closing the bootstrap (2.2c) now amounts to showing that $|V| < 2$ for all $e^{ms} \leqslant |y|$. We use (2.8) to bound the damping below

$$\mathcal{D}(y) = \beta_\tau \partial_y W \geqslant -\frac{3}{2} e^{-s}$$

In particular the damping is integrable and we obtain

$$\lambda_{\mathcal{D}} := e^{-\int_{s_*}^{s} \mathcal{D}^{y_0}\, ds'} \leqslant \varepsilon^{\frac{3}{2}}$$

Lemma 2.4.0.1 shows that the forcing is bounded and we applying our trajectory framework from Section 2.1.4.1 to obtain

$$|V^{y_0}(s)| \lesssim \lambda_{\mathcal{D}} |V(y_0)| + \lambda_{\mathcal{D}} \int_{s_*}^{s} e^{(3\alpha-1)s'}\, ds'.$$

We once again have either $s_* = s_0$ or $s_* > s_0$. In the first case our initial data assumption (2.13c) gives

$$|V^{y_0}(s)| \lesssim \lambda_{\mathcal{D}} \left(e^{s_0} \varepsilon^2 2 + e^{(3\alpha-1)s_*} \right) \leqslant 1,$$

upon ε sufficiently small. In the second case our bootstrap on $h \leqslant |y| \leqslant e^{ms}$ (2.2b) implies that

$$|V^{y_0}(s)| \lesssim \lambda_{\mathcal{D}} \left(\langle e^{ms_*} \rangle^{-2/3} + \varepsilon^{\frac{3}{4}\ell} \langle e^{ms} \rangle^{-2/3} + e^{(3\alpha-1)s_*} \right) \leqslant 1,$$

since $e^{-ms_*} \leqslant e^{-ms_0} = \varepsilon^m$. The inequality holds upon choosing ε sufficiently small. This closes the bootstrap (2.2c).

2.5. Proof of Theorem

We are now ready to prove Theorem 2.2.1 and the Corollary 2.2.1. We prove uniform Hölder bounds on our solution up to the shock time in subsection 2.5.1, and asymptotic convergence to a stable Burgers profile Ψ_ν in subsection 2.5.2. We prove Theorem 2.2.1 in subsection 2.5.3 and we conclude with subsection 2.5.5 discussing why our method cannot be extended beyond the range $0 < \alpha < 1/3$.

2.5.1. Hölder Bounds. First we note that for all $0 \leqslant |y| \leqslant h < 1/2$, the bootstrap (2.2a) gives

$$|\widetilde{W}| \leqslant \left(\varepsilon^{\frac{8}{9}(1-3\alpha)} + \log M \varepsilon^{\frac{7}{9}(1-3\alpha)}\right) h^3 \int_0^{|y|} 1\, dy = Ch^4|y| \leqslant Ch^4|y|^{1/3}.$$

Combining this estimate with the bootstrap (2.1b) we get

$$\begin{aligned} |\widetilde{W}| &\leqslant \begin{cases} Ch^3|y|^{1/3}, & 0 \leqslant |y| \leqslant h \\ \varepsilon^q \langle y \rangle^{1/3}, & h \leqslant |y| < \infty \end{cases} \\ &\leqslant |y|^{1/3}. \end{aligned}$$

We then note that the bound (2.13) for Ψ implies that

$$|W| \leqslant |\widetilde{W}| + |\Psi| \leqslant 2|y|^{1/3}. \tag{2.1}$$

From the transformation (2.3) we obtain the following equality for the Hölder seminorms of u and W

$$\begin{aligned} [u]_{C^{1/3}} = \sup_{\substack{x,z\in \boldsymbol{R} \\ x\neq z}} \frac{|u(x,t)-u(z,t)|}{|x-z|^{1/3}} &= \sup_{\substack{x,z\in \boldsymbol{R} \\ x\neq z}} \frac{e^{-s/2}|W(xe^{3s/2},s) - W(ze^{3s/2},s)|}{(e^{-3s/2}|xe^{3s/2} - ze^{3s/2}|)^{1/3}} \\ &= \sup_{\substack{x',z'\in \boldsymbol{R} \\ x'\neq z'}} \frac{|W(x',s)-W(z',s)|}{|x'-z'|^{1/3}} = [W]_{C^{1/3}}. \end{aligned}$$

Furthermore, (2.1) implies that $[W]_{C^{1/3}} \leqslant 2$ uniformly in both x and s, and since $u \in L^\infty$ we have that

$$\|u\|_{C^{1/3}} = \|u\|_{L^\infty} + [u]_{C^{1/3}} \leqslant M + 2. \tag{2.2}$$

Therefore u is Hölder $1/3$ uniformly in x and t. We also point out that by a similar argument, any Hölder norm larger than $1/3$ cannot be uniformly controlled in time and blows up when the singularity forms. Indeed, letting $\beta > 1/3$, a simple calculation shows that

$$[u]_{C^\beta} = e^{\frac{s}{2}(3\beta-1)}[W]_{C^\beta}.$$

Hence u is not C^β at T_* for any $\beta > \frac{1}{3}$.

2.5.2. Asymptotic Convergence to Stationary Solution. The key observation underpinning our analysis is that whenever $(-\Delta)^\alpha W$ is bounded the self-similar equation (2.4) governing the evolution of W formally converges to the self-similar Burgers equation (2.7). This section justifies this limit rigorously.

We follow the proof outlined by Yang in [**112**], which is based on the proof in [**15**].

2.5.2.1. *Taylor Expansion.* Set $\nu = \lim_{s\to\infty} \partial_y^3 W(0,s)$; this limit exists by the fundamental theorem of calculus and the estimate (2.6). By (2.5), we know that $5 \leqslant |\nu| \leqslant 7$. Since Ψ_ν satisfies the same constraints at the origin as Ψ, namely (2.6) (c.f. Section 2.1.4.3), we automatically obtain $W(0,s) = \Psi_\nu(0)$, $\partial_y W(0,s) = \Psi_\nu'(0)$, and $\partial_y^2 W(0,s) = \Psi_\nu''(0)$ for all $s \geqslant s_0$.

The difference $\widetilde{W}_\nu := W - \Psi_\nu$ (Ψ_ν defined in (2.12)) satisfies the evolution equation

$$\left(\partial_s - \frac{1}{2} + \Psi_\nu'\right)\widetilde{W}_\nu + \left(\frac{3}{2}y + W\right)\partial_y \widetilde{W}_\nu = -\beta_\tau F_{\widetilde{W}_\nu}, \tag{2.3}$$

where the forcing $F_{\widetilde{W}_\nu}$ is given below by (2.11). We will now show that $W(y,s) \to \Psi_\nu(y)$ for all $y \in \boldsymbol{R}$ as $s \to \infty$, that is

$$\limsup_{s\to\infty} |\widetilde{W}_\nu(y,s)| = 0. \tag{2.4}$$

The convergence (2.4) is trivial when $y_0 = 0$ since $\Psi_\nu(0) = \Psi(0)$ and $W(0,s) = 0$ by our constraint (2.6).

We consider the Taylor expansion of $\widetilde{W}_\nu$, observing that this Taylor expansion vanishes to third order as a consequence of our constraints (2.6):

$$\begin{aligned}\widetilde{W}_\nu(y,s) &= \widetilde{W}_\nu(0,s) + y\partial_y \widetilde{W}_\nu(0,s) + \frac{y^2}{2}\partial_y^2 \widetilde{W}_\nu(0,s) + \frac{y^3}{6}\partial_y^3 \widetilde{W}_\nu(0,s) + \frac{y^4}{24}\partial_y^4 \widetilde{W}_\nu(\eta,s)\\ &= \frac{y^3}{6}\partial_y^3 \widetilde{W}_\nu(0,s) + \frac{y^4}{24}\partial_y^4 \widetilde{W}_\nu(\eta,s), \qquad 0 < |\eta| < \infty.\end{aligned}$$

Differentiating (2.12) four times, applying (2.18b), and taking ε small gives the following estimate

$$\begin{aligned}\|\partial_y^4 \widetilde{W}_\nu\|_{L^\infty_{y,s}} &\leqslant \|\Psi_\nu^{(4)}\|_{L^\infty} + \|\partial_y^4 W\|_{L^\infty_{y,s}}\\ &\leqslant 30\left(\frac{7}{6}\right)^{\frac{3}{2}} + M^{\frac{6}{7}} \leqslant 2M^{\frac{6}{7}}.\end{aligned}$$

We take the absolute value of the Taylor expansion and apply (2.6) to find that

$$|\widetilde{W}_\nu(y,s)| \leqslant \frac{1}{6}|y|^3 e^{-\frac{1}{4}(3\alpha-1)s} + \frac{M^{\frac{1}{2}}}{12}|y|^4 \tag{2.5}$$

holds for all $y \in \boldsymbol{R}$.

Now fix any $y_0 \in \boldsymbol{R}$, with $|y_0| > 0$, and choose $0 < \lambda < 1$ such that $\lambda \leqslant |y_0| \leqslant \lambda^{-1/6}$. From the Taylor expansion (2.5), we may choose

$$s_* = \max\left\{\log\left(\left(\frac{6\delta}{|\lambda|^3}\right)^{-\frac{4}{3\alpha-1}}\right), s_0\right\}.$$

If we then choose some δ such that $\lambda^4 > \delta > 0$, then our Taylor expansion (2.5) collapses to

$$|\widetilde{W}_\nu(y,s)| \leqslant \delta + \frac{M^{\frac{1}{2}}}{12}|y|^4. \tag{2.6}$$

This choice for δ will become clear below.

2.5.2.2. *Lagrangian Trajectories.* Consider the Lagrangian flow associated with (2.3), given by

$$\frac{d}{ds}\Phi^{y_0} = \frac{3}{2}\Phi^{y_0} + W^{y_0}(s). \tag{2.7}$$

From the Taylor expansion of W about $y = 0$, the mean value theorem together with the uniform L^∞ bound (2.7) shows

$$|W(y,s)| \leqslant |y| \ \text{ for all } \ y \in \boldsymbol{R}.$$

Therefore $\frac{d}{ds}|\Phi^{y_0}|^2 \geqslant |\Phi^{y_0}|^2$, which upon integration yields the lower bound

$$|\Phi^{y_0}(s)| \geqslant |y_0|e^{\frac{1}{2}(s-s_0)}.$$

The same upper bound as before, (2.21), still applies to the current trajectories. Thus we have established that for all $0 < y_0 < \infty$

$$|y_0|e^{\frac{1}{2}(s-s_0)} \leqslant \Phi^{y_0}(s) \leqslant \left(|y_0| + \frac{3}{2}C\varepsilon^{-1/2}\right)e^{\frac{3}{2}(s-s_0)}. \tag{2.8}$$

Note that our new bounds are independent of any fixed parameter h, and indeed hold for any $y_0 > 0$. This is in contrast to our previous estimates, where some trajectories near the origin will not take off at all!

2.5.2.3. *Forcing Terms.* We prove two technical lemmas to deal with the fractional Laplacian in the range $1/4 \leqslant \alpha < 1/3$.

Lemma 2.5.0.1. *For $\alpha > 1/6$, we have*

$$|((-\Delta)^\alpha W)^0| \lesssim \frac{1}{h^{2\alpha}}. \tag{2.9}$$

Proof. Let χ be a smooth cutoff function such that $\chi \equiv 1$ on $0 < |y| < h/2$, $\chi \equiv 0$ on $h < |y|$, and which smoothly interpolates between 1 and 0 on $h/2 < |y| < h$ with $|\chi'| \leqslant 4/h$. We decompose

$$((-\Delta)^\alpha W)^0 = ((-\Delta)^\alpha(\chi W))^0 + ((-\Delta)^\alpha((1-\chi)W))^0$$

To estimate the first term, we use our interpolation (2.6) on $(-\Delta)^\alpha$, the bootstraps (2.1a) and (2.2a), and $|\chi'| \leqslant 4/h$ to get

$$\|(-\Delta)^\alpha(\chi W)^0\|_{L^\infty} \lesssim \|\chi W^0\|_{L^\infty}^{1-2\alpha} \|(\chi W)'\|_{L^\infty}^{2\alpha} \lesssim \frac{1}{h^{2\alpha}}.$$

To bound the second term, we use (2.2) and the fact that $2/3 + 2\alpha > 1$ to obtain

$$|(-\Delta)^\alpha[(1-\chi)W]^0| \leqslant 2C_\alpha \int_{h/2}^\infty (1-\chi(\eta)) \frac{|W(\eta)|}{\eta^{1+2\alpha}}\, dy \lesssim \int_{h/2}^\infty \eta^{-\frac{2}{3}-2\alpha}\, dy \lesssim h^{\frac{1}{3}-2\alpha}.$$

Since $h < 1$ the result follows. ■

Lemma 2.5.0.2. *For $\alpha \in (0, 1/3)$, we have*

$$\|(-\Delta)^\alpha W\|_{L^\infty} \lesssim \begin{cases} 1 + Me^{(\frac{1}{2}-3\alpha)s} & \alpha \neq 1/6 \\ M + s & \alpha = 1/6 \end{cases} \tag{2.10}$$

Proof. We decompose the domain of the singular integral representation (2.2) of $(-\Delta)^\alpha$ into three regions and estimate

$$|(-\Delta)^\alpha W(y)| \lesssim \int_{\mathbb{R}} \frac{|W(y)-W(\eta)|}{|y-\eta|^{1+2\alpha}}\, d\eta \lesssim \left(\int_{0<|\eta-y|<1} + \int_{1<|\eta-y|<e^{3s/2}} + \int_{e^{3s/2}<|\eta-y|} \right) \frac{|W(y)-W(\eta)|}{|y-\eta|^{1+2\alpha}}\, d\eta.$$

Around y, the mean value theorem and (2.7) give $|W(y) - W(\eta)| \leqslant |y-\eta|$. Therefore

$$\int_{0<|\eta-y|<1} \frac{|W(y)-W(\eta)|}{|y-\eta|^{1+2\alpha}} d\eta \lesssim \int_{0<|\eta-y|<1} \frac{1}{|y-\eta|^{2\alpha}}\, d\eta \lesssim 1.$$

In the second region we use the *a posteriori* Hölder $C^{1/3}$ seminorm (2.1) to obtain

$$\int_{1<|\eta-y|<e^{3s/2}} \frac{|W(y)-W(\eta)|}{|y-\eta|^{1+2\alpha}}\, d\eta \lesssim \int_{1<|\eta-y|<e^{3s/2}} \frac{|y-\eta|^{1/3}}{|y-\eta|^{1+2\alpha}}\, d\eta \lesssim \begin{cases} 1 + e^{(\frac{1}{2}-3\alpha)s} & \alpha \neq 1/6 \\ 1 + s & \alpha = 1/6. \end{cases}$$

Finally, we use (2.10) and (2.9) to estimate the third integral

$$\int_{e^{3s/2}<|\eta-y|} \frac{|W(y)-W(\eta)|}{|y-\eta|^{1+2\alpha}}\, d\eta \lesssim Me^{s/2} \int_{e^{3s/2}<|\eta-y|} \frac{1}{|y-\eta|^{1+2\alpha}}\, d\eta \lesssim Me^{(\frac{1}{2}-3\alpha)s}.$$

The three estimates above prove the lemma. ■

We proceed to bound the forcing term in (2.3), which is given by

$$F_{\widetilde{W}_\nu} = \underbrace{e^{-s/2}\dot{\kappa}}_{I} + \underbrace{e^{(3\alpha-1)s}(-\Delta)^\alpha W}_{II} + \underbrace{\partial_y W e^{s/2}(\kappa - \dot{\xi})}_{III} + \underbrace{\dot{\tau} W \partial_y W}_{IV} \tag{2.11}$$

In what should now feel like a *danse familière*, we go term by term bounding the forcing in L^∞.

When $\alpha < 1/4$, we use the identity (2.7b), the interpolation estimate (2.6), and the bootstrap (2.5) to obtain

$$\begin{aligned}|I| &\leqslant e^{(3\alpha-1)s}\left(\frac{|(-\Delta)^\alpha \partial_y^2 W^0(s)|}{|\partial_y^3 W^0(s)|} + |(-\Delta)^\alpha W^0(s)|\right)\\ &\lesssim M^{1/6+3\alpha} e^{(2\alpha-\frac{1}{2})s}\\ &\leqslant e^{(\alpha-\frac{1}{4})s}\end{aligned}$$

On the other hand, when $\alpha \geqslant 1/4$, we use Lemma 2.5.0.1 to handle the second term, and obtain

$$|I| \lesssim \frac{M^{1/6+3\alpha}}{h^{2\alpha}} e^{(3\alpha-1)s} \leqslant e^{\frac{1}{2}(3\alpha-1)s}.$$

Lemma 2.5.0.2 means that the second term is bounded by

$$|II| \lesssim e^{(3\alpha-1)s}\left(1 + Me^{(1/2-3\alpha)s}\right) \leqslant e^{-\frac{1}{4}s},$$

upon taking ε small.

The identity (2.7b), the interpolation estimate (2.6), the bounds (2.18c)-(2.17), the bootstrap (2.2), and the lower bound (2.8) on the trajectories gives us

$$|III| \lesssim (1+\varepsilon^{\frac{3}{4}\ell})\langle y\rangle^{-2/3} e^{(3\alpha-1)s} M^{\frac{2}{7}(1-\alpha)} \lesssim (1+\varepsilon^{\frac{3}{4}\ell}) M^{\frac{2}{7}(1-\alpha)} e^{(3\alpha-\frac{4}{3})s}.$$

For the last term, the bootstraps (2.1)-(2.2), the lower bound (2.8) on our trajectories, (2.6b), and taking ε small gives us the bound

$$|IV| \leqslant (1+\varepsilon^{\frac{1}{2}q})(1+\varepsilon^{\frac{3}{4}\ell})\langle y\rangle^{-\frac{1}{3}} \leqslant 2e^{-\frac{1}{6}s}.$$

To summarize, we have shown that

$$|F^\lambda_{\widetilde{W}_\nu}(s)| \lesssim e^{-ps}, \qquad p = p(\alpha) > 0. \tag{2.12}$$

2.5.2.4. *Putting it all Together.* We set $G(y,s) = e^{-\frac{3}{2}(s-s_*)}\widetilde{W}_\nu(y,s)$, then compose with trajectories (2.7) and compute

$$\begin{aligned}\frac{d}{ds}G^\lambda &= -\frac{3}{2}G^\lambda + e^{-\frac{3}{2}(s-s_*)}\frac{d}{ds}\widetilde{W}^\lambda_\nu\\ &= -\frac{3}{2}G^\lambda + e^{-\frac{3}{2}(s-s_*)}\left[\left(\frac{1}{2} - (\Psi'_\nu)^\lambda\right)\widetilde{W}^\lambda_\nu - \beta_\tau F^\lambda_{\widetilde{W}_\nu}\right]\\ &= \left(-1 - (\Psi'_\nu)^\lambda\right)G^\lambda - \beta_\tau e^{-\frac{3}{2}(s-s_*)} F^\lambda_{\widetilde{W}_\nu}.\end{aligned}$$

Note that the damping $1 + (\Psi'_\nu)^\lambda \geqslant 0$ for all $s \geqslant s_*$, since $\|\Psi'_\nu\|_{L^\infty} = 1$ by (2.12). Thus we can apply Grönwall's inequality, our forcing estimate (2.12), and our Taylor expansion (2.6), and take s_* sufficiently large to obtain

$$\begin{aligned}
|G^\lambda| &\leqslant |G(\lambda, s_*)| + \beta_\tau \int_{s_*}^{s} e^{-\frac{3}{2}(s'-s_*)} |F^\lambda_{\widetilde{W}_\nu}(s')|\, ds' \\
&\lesssim |\widetilde{W}^\lambda_\nu(s_*)| + 2\int_{s_*}^{s} e^{-\frac{3}{2}(s'-s_*)} e^{-ps'}\, ds' \\
&\lesssim |\widetilde{W}^\lambda_\nu(s_*)| + e^{-ps_*} \\
&\leqslant \delta + \frac{M^{\frac{1}{2}}}{12}|\lambda|^4 + \delta \\
&\lesssim M^{\frac{1}{2}}|\lambda|^4.
\end{aligned}$$

For all times $s_* \leqslant s \leqslant s_* + \frac{7}{3}\log|\lambda|^{-1}$, by our definition of G, we have that

$$\begin{aligned}
|\widetilde{W}^{y_0}_\nu| &\lesssim M^{\frac{1}{2}}|\lambda|^4 e^{\frac{3}{2}(s-s_*)} \\
&\lesssim M^{\frac{1}{2}}|\lambda|^{1/2}.
\end{aligned}$$

For all y between λ and $\Phi(\lambda, s_* + \frac{7}{3}\log|\lambda|^{-1})$, there exists $s_* \leqslant s \leqslant s_* + \frac{7}{3}\log|\lambda|^{-1}$ such that $y = \Phi(\lambda, s)$. Therefore, for any pair (y, s) the previous estimate gives

$$|\widetilde{W}_\nu(y, s)| \lesssim M^{\frac{1}{2}}|\lambda|^{1/2}.$$

By composing with our trajectory lower bound this will cover at least all y such that

$$\lambda \leqslant |y| \leqslant \lambda e^{\frac{1}{2}(s-s_*)} = \lambda^{-1/6}.$$

Now, taking the limit that $s_* \to \infty$, for all $\lambda \leqslant |y| \leqslant \lambda^{-1/6}$ we have that

$$\limsup_{s\to\infty} |\widetilde{W}_\nu(y, s)| \lesssim M^{\frac{1}{2}}|\lambda|^{1/2}.$$

Finally, taking $\lambda \to 0$ proves that for all $y \neq 0$

$$\limsup_{s\to\infty} |\widetilde{W}_\nu(y, s)| = 0.$$

This completes the proof of (2.4).

2.5.3. Proof of Theorem 2.2.1. We are finally equipped to prove our main theorem, Theorem 2.2.1.

(i) **(Solution is smooth before T_*).** Recall the uniform L^2 bound for $\partial_y W$ proven in Lemma 2.3.0.1 and the uniform L^2 bound of $\partial_y^6 W$ proven in (2.16); interpolation via Gagliardo-Nirenberg (Lemma 2.1.1.2) bounds the intermediary L^2 norms of W uniformly in self-similar time.

From the transformation (2.3) and by differentiating through the L^2 norm we obtain the following family of identities

$$\|\partial_x^n u\|_{L^2} = e^{\left(-\frac{5}{4}+\frac{3}{2}n\right)s}\|\partial_y^n W\|_{L^2}$$

relating the L^2 norms in physical space to the L^2 norms in the self-similar space. Note that Burgers equation satisfies L^2 conservation and the fractional term may be dropped from estimates, ensuring that $\|u\|_{L^2}$ is uniformly bounded. These identities along with the uniform boundedness of the L^2 norms for $\partial_y W$ through $\partial_y^6 W$ shows that the H^6 norm of u remains finite for all times prior to T^*.

From the above considerations and the local-in-time existence proven in [**2**,**56**] we deduce that $u \in C([-\varepsilon, \overline{T}]; H^6(\boldsymbol{R}))$ for any $\overline{T} < T_*$.

(ii) **(Blowup location is unique).** Fix $x^\flat \neq x_*$. Differentiating the transformation (2.3) gives the identity

$$\partial_x u(x^\flat, t) = e^s \partial_y W\left((x^\flat - \xi(t))e^{\frac{3}{2}s}, s\right).$$

Because $x^\flat \neq x_*$ and $\xi(t) \to x_*$ as $t \to T_*$, there exists a time $t^\flat$ such that for all $t^\flat < t \leqslant T_*$ we can choose $1 > \lambda > 0$ such that

$$|x^\flat - \xi(t)| > \lambda > 0.$$

Therefore

$$|x^\flat - \xi(t)|e^{\frac{3}{2}s} > \lambda e^{\frac{3}{2}s},$$

and for all

$$s > s_\lambda := \max\{\frac{3}{2}\left(\frac{1}{3/2 - m}\right)\log\lambda^{-1}, -\log(T_* - t^\flat)\},$$

we satisfy the inequality

$$|x^\flat - \xi(t)|e^{\frac{3}{2}s} > e^{ms}.$$

We emphasize that the lower bound s_λ on times for which the above inequality holds grows arbitrarily large as $\lambda \to 0^+$.

We can apply our bootstrap (2.2c) to find that for all s in this range

$$|\partial_y W\left((x^\flat - \xi(t))e^{\frac{3}{2}s}, s\right)| \lesssim e^{-s},$$

and hence

$$\limsup_{t\to T_*} |\partial_x u(x^\flat, t)| \lesssim 1.$$

Thus the blowup location is unique.

(iii) **(Blowup time and location).** From our definition of τ, the blowup time T_* is the unique fixed point $\tau(T_*) = T_*$, and in light of (2.1) this is equivalent to

$$\int_{-\varepsilon}^{T_*} (1 - \dot{\tau}(t))\, dt = \varepsilon.$$

Applying the bootstrap closure (2.1) we find that

$$\varepsilon = \int_{-\varepsilon}^{T_*} (1 - \dot{\tau}(t))\, dt \geqslant \int_{-\varepsilon}^{T_*} 1 - \frac{3}{4}\varepsilon^{\frac{8}{9}(1-3\alpha)}\, dt,$$

from which it follows that $|T_*| \leqslant \frac{3}{4}\varepsilon^{\frac{8}{9}(1-3\alpha)}$ upon choosing ε sufficiently small.

Similarly, our definition of ξ together with (2.1) gives the condition

$$\int_{-\varepsilon}^{T_*} \dot{\xi}(t)\, dt = x_*.$$

Applying the bootstrap closure (2.2) gives us

$$x_* = \int_{-\varepsilon}^{T_*} \dot{\xi}(t)\, dt \leqslant 2M(T_* + \varepsilon) \leqslant 3M\varepsilon.$$

This proves the claimed bounds on T_* and y_*.

(iv) **(Precise control of $\partial_x u$ at $t = T_*$).** u develops a shock (gradient blowup) at $t = T_*$. Recall the identity (2.2) which, together with differentiating the transformation (2.3), gives

$$\partial_x u(\xi(x), t) = \frac{1}{\tau(t) - t}\partial_y W(0, s) \geqslant -\frac{1}{\tau(t) - t}.$$

Next note that for all $-\varepsilon \leqslant t < T_*$ we have that

$$\frac{1}{2} \leqslant \frac{\tau(t) - t}{T_* - t} \leqslant 1.$$

The upper bound is obvious since $\tau(t)$ is monotone increasing and T_* is the unique fixed point of τ. The lower bound is a consequence of the fact that

$$\frac{T_*}{2} \leqslant \tau(t) - \frac{1}{2}t.$$

This follows from the fact that $\tau(t) - t/2$ is monotone decreasing and at time $t = T_*$ the r.h.s. is $T_*/2$. We conclude that

$$-\frac{1}{\tau(t)-t} \leqslant \partial_x u(\xi(x), t) = -\|\partial_x u(\cdot, t)\|_{L^\infty} \leqslant -\frac{1}{2}\frac{1}{\tau(t)-t}.$$

Thus we have proven the desired behavior of the gradient at the singularity.

(v) **($W \to \Psi$ asymptotically in self-similar space).** This was proved above in Section 2.5.2.

(vi) **(Shock is Hölder 1/3).** This is a consequence of the uniformity of the Hölder bound which was shown above in Section 2.5.1. Take the limit as $s \to \infty$ to find that $u(y, T_*) \in C^{1/3}$.

2.5.4. Proof of Corollary 2.2.1. We begin by noting that κ_0 and ε can be taken in an open neighborhood since all of our previous arguments require only that ε is sufficiently small. From (2.10), this implies that $u_0(0)$ and $\partial_x u_0(0)$ can be taken in an open set.

Interpolating between (2.21b) and (2.21c) yields $\|\partial_x^4 u_0\|_{L^\infty} = \mathcal{O}(\varepsilon^{-11/2})$. Using this in a Taylor expansion for $\partial_x^2 u_0$ around the origin, we find that for all $0 < |\overline{x}| < |x|$

$$\begin{aligned}\partial_x^2 u_0(x) &= \partial_x^2 u_0(0) + \partial_x^3 u_0(0)x + \frac{\partial_x^4 u_0(\overline{x})}{2}\overline{x}^2 \\ &= \partial_x^2 u_0(0) + 6\varepsilon^{-4}x + x(\partial_x^3 u_0(0) - 6\varepsilon^{-4}) + \mathcal{O}(\varepsilon^{-11/2})\overline{x}^2.\end{aligned}$$

We will show that we can relax $\partial_x^2 u_0(0) = 0$ and (2.17) by assuming instead that $|\partial_x^2 u_0(0)|$ is sufficiently small, and considering a small interval around 0 for x on which $|\partial_x^3 u_0(x) - 6\varepsilon^{-4}| < 7\varepsilon^{-4/9(5+12\alpha)}$. We can take $x(\partial_x^3 u_0(0) - 6\varepsilon^{-4})$ small relative to $6\varepsilon^{-4}x$, and similarly for $\mathcal{O}(\varepsilon^{-11/2})\overline{x}^2$. Since $6\varepsilon^{-4}x$ dominates, by intermediate value theorem, there exists x^* sufficiently small so $\partial_x^2 u_0(x^*) = 0$. We can now simply change coordinates $x \mapsto x + x^*$.

The inequalities in (2.18)-(2.21c) can be replaced by strict inequalities by introducing a pre-factor slightly greater than 1. A sufficiently small H^6 perturbation preserves all these inequalities by slightly enlarging the pre-factor. n particular, by Sobolev embedding, a small $W^{5,\infty}$ perturbation will satisfy these inequalities.

2.5.5. Stable Modulation Past $\alpha = 1/3$. Since the stable Burgers profile satisfies $-1 < \Psi'(x) < 0$ for $x \neq 0$ with $\Psi'(0) = -1$, it follows from (2.2) that $(-\Delta)^\alpha[\Psi'](0) < 0$. If we consider W to be a perturbation of Ψ, the modulation constraint (2.7a) for τ amounts to

$$\dot{\tau} \approx e^{(3\alpha-1)s}|(-\Delta)^{\alpha}[\Psi'](0)|. \tag{2.13}$$

When $\alpha \geqslant {}^{1}/{}_{3}$, the right-hand side is positive and bounded away from zero for all s, so the modulation parameter τ diverges as $s \to \infty$.

Oh and Pasqualotto [**77**] have recently proven that the solutions to fractal Burgers equation (2.1) in the range ${}^{1}/{}_{3} \leqslant \alpha < {}^{1}/{}_{2}$ with specifically chosen initial data converge in an asymptotically self-similar manner to *unstable* profiles. However these solutions are not stable under *generic* perturbations.

The blowup criterion supplied by Dong, Du, and Li [**33**] is continuous which means that there is stable singularity formation in the range ${}^{1}/{}_{3} \leqslant \alpha < {}^{1}/{}_{2}$. Our numerics suggest that this stable blowup profile is not the stable Burgers profile.

CHAPTER 3

Computational Efficiency: n-Tangent Prop

The content of this chapter is based on the author's previous work [20].

In the previous chapter we analyzed the behavior of smooth solutions to fractal Burgers equation and found that they asymptotically approach the self-similar shock solutions for the standard Burgers equation. In the case of fractal Burgers equation we have an exact characterization of the self-similar profiles, but for more complicated equations of interest, like the Boussinesq equation, our best access to the asymptotic profiles is through numerics.

In this chapter we study physics-informed neural networks (PINNs), which are a numerical method which has been found to be well-suited to the study of these types of problems. One disadvantage to using PINNs is that they can be prohibitively slow when solving some problems, and we introduce a method in this chapter to improve their computational efficiency.

We then use our method to compute previously intractable self-similar profiles to the Burgers equation. Our work has particular implications for equations whose high-order self-similar solutions are not known, for example when studying self-similar blow up for the Boussinesq equations [105]. The present work could eventually be extended to describe unstable self-similar blowup for these fluid equations.

Finally, we note that our method is exact, meaning that we do not compromise on model accuracy. Thus, any PINN study using sufficiently many derivatives would be accelerated by adopting our methodology.

3.1. Introduction

Physics-informed neural networks (PINNs) were introduced as a numerical method to solve forward and inverse problems involving differential equations using neural networks instead of traditional numerical solvers [84]. Their use has recently come under scrutiny for several reasons, including a lack of high-accuracy results, poor run-times compared to standard numerical methods, and complicated training dynamics [72]. Due to methodological issues in the aforementioned studies, including the failure to use the well-established strategies outlined in [101], we believe that the pursuit of PINNs should not be abandoned and to that end, propose an algorithm which partially addresses the valid concerns over PINN training times.

A primary reason to use PINNs over standard numerical methods is because by approximating the solution to an ODE or PDE by a neural network we obtain a C^∞ approximation to the model solution which can be evaluated at arbitrary points in the domain, as well as allowing for the study of high-order derivatives (see Figures 3.11 and 3.14). This strength is also a weakness, since during training we must repeatedly take derivatives of the neural network with respect to the inputs. This is traditionally done using autodifferentiation [7, 84], which suffers from unfavorable scaling when taking multiple derivatives. In particular, taking n derivatives of a neural network f with M parameters gives the exponential runtime $\mathcal{O}(M^n)$ (see Theorem 3.3.1 below). For training PINNs we often need to take three or more derivatives, and in many practical applications this exponential runtime already becomes prohibitive. Furthermore this difficulty cannot simply be overcome by horizontally scaling compute, since repeated applications of autodifferentiation cannot be parallelized on a GPU due to the recursive nature of computing high-order derivatives.

In this paper we introduce n-TP, which addresses these issues by computing $\frac{d^n}{dx^n}f(x)$ in quasilinear $\mathcal{O}(e^{\sqrt{n}}M)$ time instead of the exponential time $\mathcal{O}(\frac{e^{\sqrt{n}}}{n}M^n)$. n-TP is an exact method, and thus there is no accuracy degradation when using this method. n-TP is the natural extension of the TP formalism [91] to n derivatives. TP was introduced in the context of MNIST digit classification as a way to enforce a smoothness condition on the first derivative of a neural network based classifier. The motivation for n-TP starts from the observation that for PINN training we only need higher-order derivatives with respect to the network inputs, not with respect to the network weights. In this case, naively applying autograd results in an exponential blowup in computation which slows down any PINN training. Our method constructs the minimal computation graph needed to compute $\frac{d^n}{dx^n}f(x)$.

Our contributions are three-fold

(1) We derive the n-TP formalism and give an algorithm for it's implementation.

(2) We show that for simple network architectures consisting of stacked linear, densely connected layers with the tanh activation function, our method empirically follows the theoretical scaling laws for a variety of widths, depths, and batch sizes.

(3) We show that in the context of PINN training our method can significantly reduce training time and memory requirements when compared to the standard PINN implementation, allowing us to compute previously intractable profiles.

3.2. Physics Informed Neural Networks

3.2.1. Solving Forward Problems with PINNs. PINNs are neural networks trained to approximate the solution to a given ODE or PDE [84]. Because neural networks with smooth activation functions are C^∞ function approximators, and C^∞ functions are dense in most function space which are used in practice (like the L^2 based Sobolev family of function spaces), we can train a neural network using gradient descent to approximate the solution to a given differential equation. It is this simple observation that led to the introduction of PINNs in the paper [84]. We give a brief overview of the methodology behind PINNs, but refer the reader to the recent surveys [3, 42, 100] and the references therein, as the literature abounds with introductory material on PINNs.

Let $u_\theta(\boldsymbol{x})$ be a feed-forward, densely connected, neural network with parameters θ. Our goal is to optimize these parameters in such a way that u_θ is an approximate C^∞ solution to the differential equation $F(\partial^\alpha u; \boldsymbol{x}) = 0$ for some multi-index α [40]. We train the neural network on the discrete domain $\Omega = \{\boldsymbol{x}_1, \cdots \boldsymbol{x}_N\}$ using the loss target

$$L(u_\theta) = \frac{1}{N}\sum_{k=1}^{N} |F(\partial^\alpha u_\theta; \boldsymbol{x}_k)|^2 + \text{BC}, \tag{3.1}$$

which is the mean-squared error (MSE) of the differential equation residual with BC being appropriately enforced boundary conditions. Since u_θ is $|\alpha|$-times continuously differentiable we can use autodifferentiation to exactly compute the derivatives appearing above, and thus by the approximation theorem [25, 50], if the solution to $F(\partial^\alpha u; \boldsymbol{x}) = 0$ lies in a space in which C^∞ functions are dense, we can theoretically train a neural network to approximate the true solution of the PDE to arbitrary precision.

In practice this is not so easy: PINN training is complicated by needing to enforce boundary conditions, which introduces problems inherent to multi-target machine learning [11, 102]. There is also the problem of effectively choosing collocation points from the domain, as well as the problem of choosing a good network architecture [101, 114]. PINN training appears naturally unstable, likely due to a poorly conditioned Hessian, and thus training these networks can be difficult [58, 86, 103]. This instability has further been related to the conditioning of a specific differential operator related to the underlying differential operator [29]. PINNs also appear to struggle fitting high-frequency components of the target solution [71] as a consequence of spectral bias (f-principle) [83, 111].

Additionally, convergence under the loss function (3.1) is often slow. In practice it is usually better to use "Sobolev training" [26, 69, 92] which replaces (3.1) with the Sobolev-norm [40] loss function

$$L^{(m)}_{\text{Sobolev}}(u_\theta) = \frac{1}{N}\sum_{k=1}^{N}\sum_{j=0}^{m} Q_j |\nabla^j_{\boldsymbol{x}} F(\partial^\alpha u_\theta; \boldsymbol{x}_k)|^2 + \text{BC}, \tag{3.2}$$

where Q_j are relative weights which add additional hyperparameters to the training [69]. While this loss function generally improves accuracy, it also requires computing m extra derivatives of the neural network u_θ. Due to the nature of autodifferentiation, this trade-off quickly becomes costly and in practice we can often only train with $m = 1$ or $m = 2$, despite the fact that higher m usually results in better solution accuracy. n-TP makes this trade-off much cheaper and we hope that future authors are able to train with $m = 4$ or higher while retaining reasonable training times.

The appearance of high-order derivatives is also not uncommon in PINN applications. Wang et al. [105] show that to satisfactorily compute successive high-order unstable shock profiles for the Burgers, De Gregorio, and Boussinesq equations, one must take high-order derivatives. For example, to compute the m-th smooth, self-similar shock profile for Burgers equation one must take $2m + 3$ derivatives. The authors compute the first and second profiles which already requires taking five derivatives and is already slow.

3.3. n-TP

3.3.1. Differentiating an MLP Network.

Consider the L-hidden layer MLP network given recursively by

$$\begin{aligned} u &= \boldsymbol{W}^{L+1}\boldsymbol{h}^L, \\ \boldsymbol{h}^\ell &= \sigma(\boldsymbol{a}^\ell), \\ \boldsymbol{a}^\ell &= \boldsymbol{W}^\ell \boldsymbol{h}^{\ell-1}, \\ \boldsymbol{a}^0 &= \boldsymbol{W}^0 \boldsymbol{x}. \end{aligned} \tag{3.1}$$

We will consider taking the n-th derivative of this MLP with respect to the input vector $\boldsymbol{x}$.

Recall that Faá di Bruno's formula [87] states that for the composition of C^n continuous functions f and g we have

$$(f(g(x)))^{(n)} = \sum_{\boldsymbol{p}\in\mathcal{P}(n)} C_{\boldsymbol{p}} \left(f^{|\boldsymbol{p}|}\right)(g(x)) \prod_{j=1}^{n} \left(g^{(j)}(x)\right)^{p_j}, \tag{3.2}$$

where the sum is taken over the set $\mathcal{P}(n)$ of partition numbers of order n, and consists of all tuples $\boldsymbol{p}$ of length n and satisfying $\sum_{j=1}^{n} jp_j = n$, $0 \leqslant p_j \leqslant n$, and $|\boldsymbol{p}| = \sum_j p_j$. The constants $C_{\boldsymbol{p}}$ are explicitly computable and the size of the set $\mathcal{P}$ is found using the partition function $p(n) = |\mathcal{P}(n)|$ whose combinatorial properties are well-studied [13].

Using Faá di Bruno's formula (3.2), we can recursively differentiate the computation done by the MLP network 3.1. Applying the chain rule and Faà di Bruno's formula gives us

$$\frac{d^n}{dx^n}u = \boldsymbol{W}^{L+1}\frac{d^n}{dx^n}\boldsymbol{h}^L, \tag{3.3a}$$

$$\frac{d^n}{dx^n}\boldsymbol{h}^\ell = \sum_{\boldsymbol{p}(n)} C_{\boldsymbol{p}}\sigma^{(|\boldsymbol{p}|)}(\boldsymbol{a}^\ell)\prod_{k=1}^{n}\left(\frac{d^k}{dx^k}(\boldsymbol{a}^\ell)\right)^{p_k}, \tag{3.3b}$$

$$\frac{d^n}{dx^n}\boldsymbol{a}^\ell = \boldsymbol{W}^\ell\frac{d^n}{dx^n}\boldsymbol{h}^{\ell-1}, \tag{3.3c}$$

$$\frac{d^n}{dx^n}\boldsymbol{h}^0 = \delta_{n=0}\boldsymbol{W}^0\boldsymbol{x} + \delta_{n=1}\boldsymbol{W}^0. \tag{3.3d}$$

3.3.2. Exponential Growth of the Autograd Graph. Autodifferentiation is a method for computing the derivatives of a neural-network using the network's computational graph [7, 80]. Autodifferentiation is usually applied in the context of gradient descent for optimizing neural networks where it is used to efficiently and exactly compute the first partial derivatives of the loss with respect to each of the network weights [80]. It is usually applied once per training iteration, and outputs a vector of first-order partial derivatives with respect to all network inputs (including the weights). It is able to do this computation in $\mathcal{O}(M + d) = \mathcal{O}(M)$ time, where M is the number of model parameters and d is the dimensionality of the input.

However, when applied repeatedly to compute the x derivatives of a neural network as is done in equation (3.3), autodifferentiation repeats computation and generates a graph which grows exponentially in the number of derivatives. Figure 3.1 demonstrates this fact empirically by plotting the number of nodes in the computational graph as a function of the number of derivatives taken.

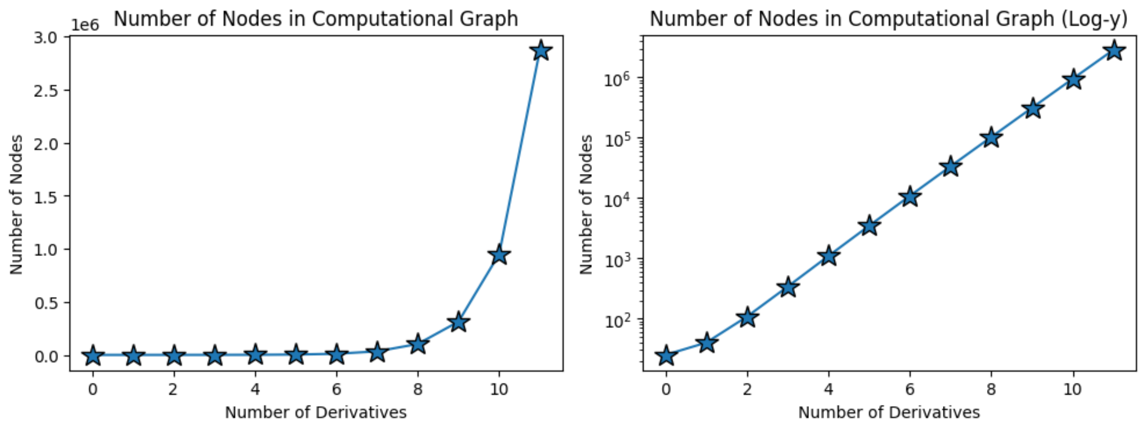


FIGURE 3.1. Number of nodes in the Autograd graph generated by Jax as a function of number of derivatives. Note the clear exponential growth in the number of nodes, consistent with our mathematical observation in Theorem 3.3.1.

We can quantify this growth theoretically as well. The following theorem tells us how quickly the number of nodes in the computational graph grows as we take successive derivatives.

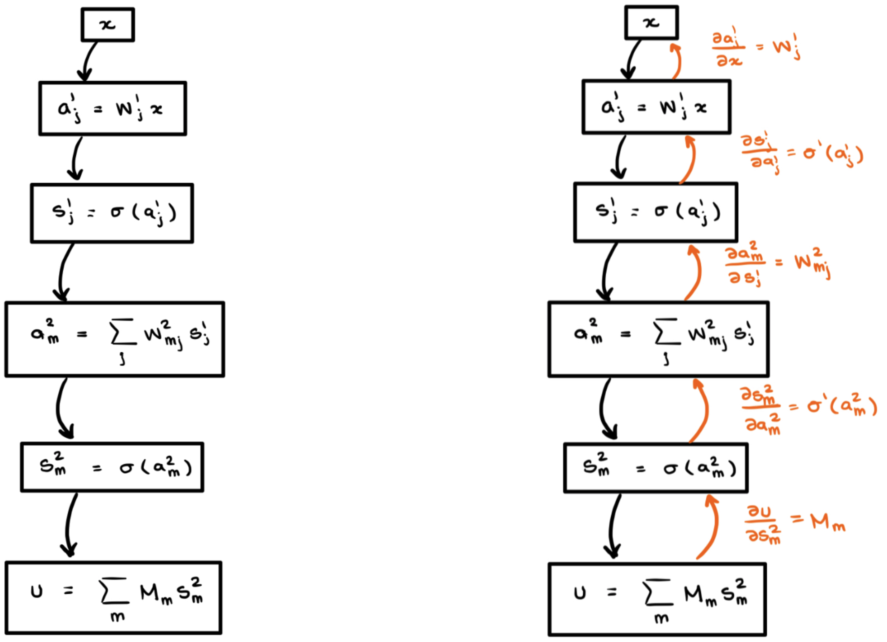

x
$a^1_j = W^1_j x$
$s^1_j = \sigma(a^1_j)$
$a^2_m = \sum_j W^2_{mj} s^1_j$
$s^2_m = \sigma(a^2_m)$
$U = \sum_m M_m s^2_m$
$\frac{\partial a^1_j}{\partial x} = W^1_j$
$\frac{\partial s^1_j}{\partial a^1_j} = \sigma'(a^1_j)$
$\frac{\partial a^2_m}{\partial s^1_j} = W^2_{mj}$
$\frac{\partial s^2_m}{\partial a^2_m} = \sigma'(a^2_m)$
$\frac{\partial U}{\partial s^2_m} = M_m$

FIGURE 3.2. **(Left):** Autograd graph for the forward pass of the zeroth derivative. **(Right):** The autograd graph for the forward pass of the zeroth derivative with the associated derivative computations which are computed during the backwards pass. These backwards values are used to construct the full autograd graph for the first derivative. Unrolling the orange edges gives us the first derivative graph shown in Figure 3.3, left.

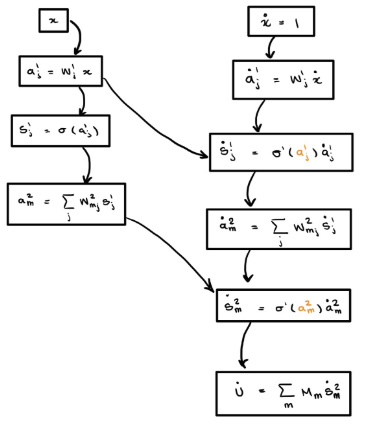
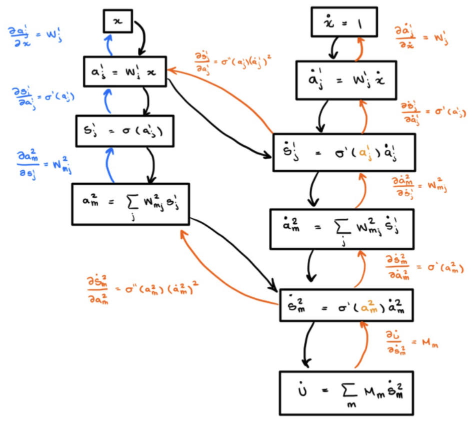



FIGURE 3.3. **(Left):** Autograd graph for the forward pass of the first derivative. **(Right):** The autograd graph for the forward pass of the first derivative with the associated derivative computations which are computed during the backwards pass. The values in blue represent repeated computation which is mathematically unnecessary for computing the second derivative autograd graph. This repeated computation is seen even more clearly when we construct the autograd graph of the second derivative in Figure 3.4.

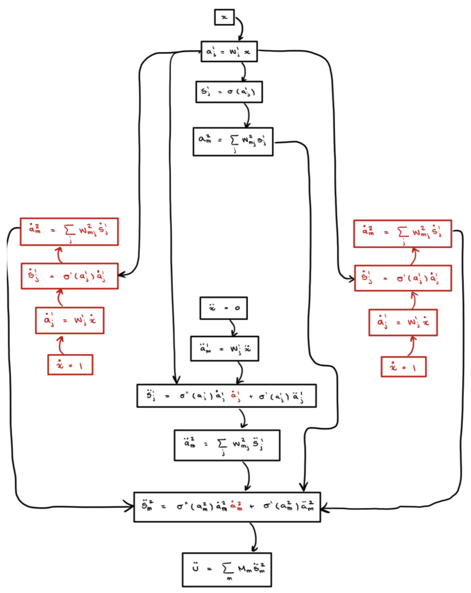


FIGURE 3.4. Full autograd graph of the second derivative. Observe that the computation for the first derivative is done twice (represented in red), introducing redundancy into the computation. This computation is unnecessary and a more minimal graph can be constructed using n-TP (see Figure 3.5).

THEOREM 3.3.1. *Let f be a densely connected feed-forward neural network of the form* (3.1)*. Then the size of the computation graph for the forward pass of the n-th derivative $f^{(n)}$ of f with respect to the network inputs x grows exponentially with the bound $\mathcal{O}(2^n)$.*

Proof of Theorem 3.3.1. Figure 3.2 shows that for an MLP network of depth L we will have $2(L+2)$ "nodes" in our computational graph. Observe from Figures 3.2 and 3.3 that taking the first derivative adds $2(L+2)$ nodes, one for each node in the original graph, and subtracts 2 nodes, corresponding to the final layers of the MLP which are not needed to compute the second derivative. Thus, the second derivative has $2(L+2)+2(L+2)-2 = 4L+6 = 4(L+1)+2$ nodes.

This pattern continues: each derivative generates the same number of nodes in the previous graph, and subtracts the two nodes corresponding to the output of the MLP. Thus, we have that the third derivative has $2(4L+6)-2 = 8L+10 = 8(L+1)+2$. This pattern continues, and after n derivatives we have

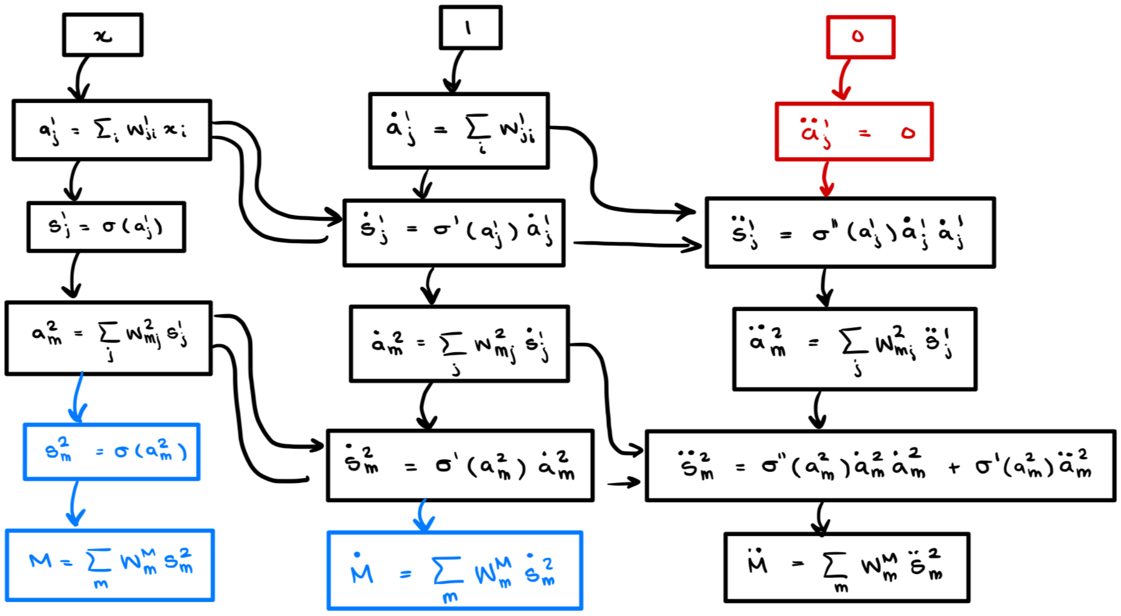
$$\#\text{nodes} = 2^n(L+1)+2. \tag{3.4}$$

■

Note that the FLOPs grow at least proportionally to the number of nodes in the computation graph, so we expect that the amount of computation being done grows concomitantly with the graph size.

3.3.3. TangentProp and *n*-TP. TangentProp is an alternative to autodifferentiation for computing the derivative of a neural network with respect to the network inputs, developed prior to autodifferentiation being widely used [91]. The TangentProp algorithm was originally developed to directly compute the first derivative of a neural network using the formula (3.3). This derivative is then used to ensure that the network outputs are invariant to small rotations and translations of the input which led in the authors' use case to a significant increase in performance.

We make the straightforward observation that TangentProp can easily be extended to an arbitrary number of derivatives using the formula we derived above in (3.3), and the less straightforward observation that doing so can be done efficiently and gives rise to a linear scaling instead of an exponential scaling. We also make the observation that n-TP gives us the minimal computational graph needed to compute the n-th derivative of a neural network (see below).



x
1
0
$a^1_j = \sum_i W^1_{ji} x_i$
$\dot a^1_j = \sum_i W^1_{ji}$
$\ddot a^1_j = 0$
$s^1_j = \sigma(a^1_j)$
$\dot s^1_j = \sigma'(a^1_j)\dot a^1_j$
$\ddot s^1_j = \sigma''(a^1_j)\dot a^1_j \dot a^1_j$
$a^2_m = \sum_j W^2_{mj} s^1_j$
$\dot a^2_m = \sum_j W^2_{mj} \dot s^1_j$
$\ddot a^2_m = \sum_j W^2_{mj} \ddot s^1_j$
$s^2_m = \sigma(a^2_m)$
$\dot s^2_m = \sigma'(a^2_m)\dot a^2_m$
$\ddot s^2_m = \sigma''(a^2_m)\dot a^2_m \dot a^2_m + \sigma'(a^2_m)\ddot a^2_m$
$M = \sum_m W^M_m s^2_m$
$\dot M = \sum_m W^M_m \dot s^2_m$
$\ddot M = \sum_m W^M_m \ddot s^2_m$

FIGURE 3.5. The autograd graph for the computation done by n-TP. Observe that there is no repeated computation and that every node is necessary, meaning that this graph is the minimal graph required to compute n derivatives.

Finally, we observe that the autograd graph generated by n-TP is minimal in the sense that it cannot be reduced further while performing the same computation. Figure 3.5 shows the full graph for $n = 2$ and we can see that there are no more nodes to prune from this graph (c.f. Figure 3.4 for the same computation). Each of the operations required for Equation (3.3) is computed only once.

Notes on the Implementation of n-TP: When we compute the derivative 3.3 using autodifferentiation, the constants $C_{\boldsymbol{p}}$ and the associated summations over $\boldsymbol{p}(n)$ are automatically and implicitly computed. However, when we directly compute 3.3 we must now explicitly account for these factors. To do so requires computing the Bell coefficients [13]. This is a standard, but expensive, computation, and we found that it is best to pre-compute the maximal number of Bell coefficients needed for the given training run, and then cache these values so that we can access them using a $\mathcal{O}(1)$ lookup table. An algorithm listing can be found in Algorithm **??**.

Runtime of n-TP and Autodiff: For the sake of completeness we give a tighter bound on the runtime which takes into account the dependence on n of the summation appearing in (3.2). The combinatorial properties of the summation over the integer partitions $\mathcal{P}(n)$ are well studied. In particular, the partition function $p(n)$ counts the number of integer partitions and thus $p(n) = |\mathcal{P}(n)|$. A well-known and classical result of Hardy and Ramanujan [48] provides an upper and lower bound on the partition function $p(n)$ which

Algorithm 1 Forward Pass with Higher-Order Derivatives

```
1:  procedure FORWARD($x, n$)                          ▷ $x$: input, $n$: derivative order
2:      if $n = 0$ then
3:          for $layer$ in NETWORK do
4:              $x \leftarrow layer(\sigma(x))$                 ▷ $\sigma$ is activation function
5:          return $x$
6:      $y \leftarrow$ array of length $n + 1$
7:      $y_0 \leftarrow L_1(x)$                               ▷ $L_1$ is first layer
8:      $y_1 \leftarrow L_1(\mathbf{1}) - b_1$
9:      for $i \leftarrow 2$ to $n$ do
10:         $y_i \leftarrow L_1(\mathbf{0}) - b_1$
11:     for $L$ in NETWORK[2 :] do
12:         $a \leftarrow \sigma(y_0, n)$
13:         for $i \leftarrow n$ down to 1 do
14:             $z \leftarrow \mathbf{0}$
15:             for $(c, e)$ in BELL[$i$] do
16:                 $s \leftarrow \sum e$
17:                 $t \leftarrow \prod_{j:e_j \neq 0} y_j^{e_j}$
18:                 $z \leftarrow z + c \cdot t \cdot a_s$
19:             $y_i \leftarrow z$
20:         $y_0 \leftarrow L(a_0)$
21:         for $i \leftarrow 1$ to $n$ do
22:             $y_i \leftarrow L(y_i) - b$                   ▷ Subtract bias
23:     return $y$
```

then yields the asymptotic behavior

$$p(n) = \mathcal{O}\left(\frac{e^{\sqrt{n}}}{n}\right).$$

This implies the more refined runtime depending on n and M of $\mathcal{O}(np(n)M) \sim \mathcal{O}\left(e^{\sqrt{n}}M\right)$ which is our claimed quasilinear bound. Note that during autodifferentiation the Faà di Bruno formula must implicitly be applied to the activation function σ and therefore autodifferentiation has an actual asymptotic runtime of $\mathcal{O}\left(\frac{e^{\sqrt{n}}}{n}M^n\right)$. Finally we remark that the memory complexity of n-TP is linear at $\mathcal{O}(nM)$ (without a modification from $p(n)$) while the memory complexity of autodifferentiation is exponential at $\mathcal{O}(M^n)$. Thus, not only does our algorithm compute derivatives faster than autodifferentiation, but we can compute more derivatives on the same hardware than is even possible using autodifferentiation.

3.4. Empirical Results

We now turn to an empirical validation of our theoretical algorithm proposed above. We begin by demonstrating that for a wide range of feed-forward neural network architectures that our proposed method indeed follows the theoretical asymptotic scaling. In particular we consider a standard feed-forward network with uniform width across the layers, an the `tanh` activation function. Next, we use our method to train a

PINN model to compute the smooth stable and unstable profiles for the self-similar Burgers profile using the methodology proposed in [105]. This problem requires taking a large number of derivatives for the training to converge, and we demonstrate that our method is able to break through the computational bottlenecks imposed by autodifferentiation. Using n-TP we are able to compute higher-order profiles than previous works, and we show that the computation of these profiles is infeasible on standard hardware without n-TP.

3.4.1. Implementation Details and Methodology. We run our experiments using Python and PyTorch [80]. In particular we implement n-TP as a custom forward method for a PyTorch `torch.nn.Module` implementation of a deep feed-forward network. For the PINN experiments we then build a custom PINN training framework to handle the PINN training loop, and we use an open-source L-BFGS implementation [90] instead of the PyTorch L-BFGS due to the latter not supporting line-search[1]. All experiments were run locally on a single NVIDIA A6000 GPU.

3.4.2. Forwards and Backwards Pass Times. PyTorch implements several asynchronous optimizations that make benchmarking difficult. To mitigate the effects of built-in optimizations on our benchmarking we implement the following steps

(1) Randomly shuffle the experiments over all parameters to ensure that execution order is not a factor in the results.
(2) Synchronize CUDA between runs.
(3) Enable `cudnn.benchmark`.
(4) Run the Python garbage collector between runs.
(5) Use the Python performance counter instead of the timing module.

These mitigation strategies allow for a fairer comparison between autodifferentiation and n-TP.

We first explore the effect of n-TP on the computation of a single forward and backward pass to verify that the empirical performance aligns with the predicted theoretical performance suggested by our derivation. In particular we would expect to see exponential run-times for autodifferentiation and quasilinear run-times for n-TP.

For a given network we compute and time the forward pass through the network, compute the loss outside of a timing module, then compute and time a backwards pass through the network. The total time includes the time it takes to compute the loss function, while the forward and backwards pass times only include the time it takes to compute the given pass.

[1]See for example https://discuss.pytorch.org/t/optimizer-with-line-search/19465 and https://discuss.pytorch.org/t/l-bfgs-b-and-line-search-methods-for-l-bfgs/674 for further discussion of this deficiency.

For a fixed network size, we find that the end-to-end times for a combined forward and backward pass for autodifferentiation scales exponentially and that n-TP scales roughly quasilinearly (Figures 3.6 and 3.7), keeping with the theoretical predictions made above by our formalism. We can further decompose this total execution time into its forward and backward times (Figure 3.6 and Figure 3.7 respectively). We see that the n-TP formalism gives more significant performance gains during the forward pass when compared to the backwards pass. We hypothesize that this is due to PyTorch graph optimizations that are applied automatically to the autodifferentiation implementation and are not included in our n-TP implementation. This difference is seen most plainly in Figure 3.7, where autodifferentiation outperforms n-TP in backwards pass times for small numbers of derivatives.

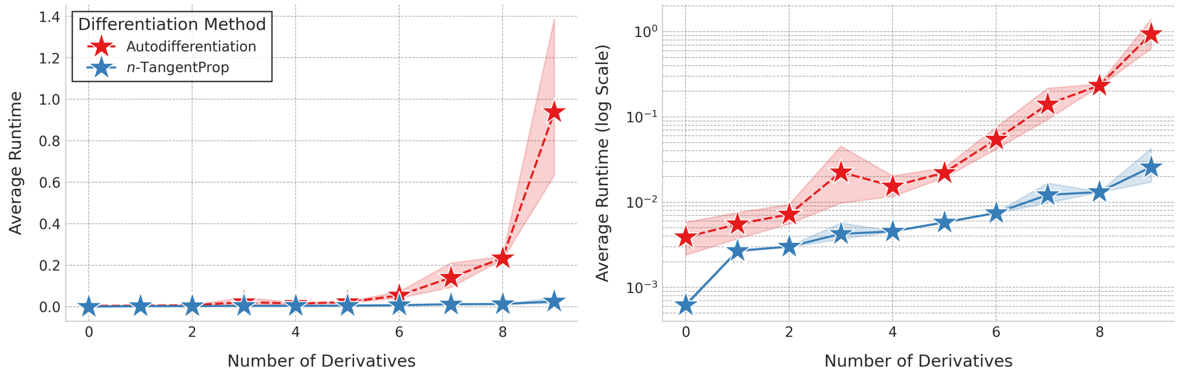


FIGURE 3.6. Forward pass times for a simple MLP network. The left and right frames show the same data, however the right frame is plotted with a logarithmic y-axis. Each model is run 100 times and the average for each trial is plotted. The network has 3 hidden layers of 24 neurons each, a common PINN architecture. The batch size is $2^8 = 256$ samples.

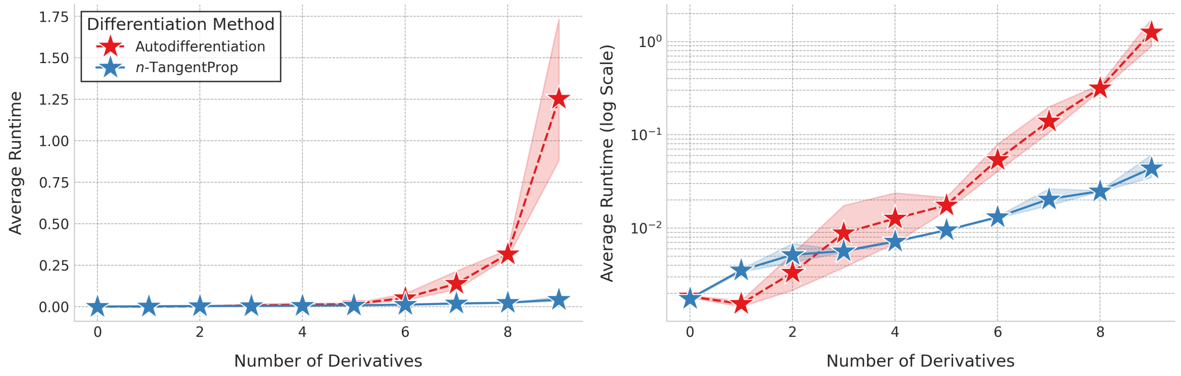


FIGURE 3.7. Backwards pass times for a simple MLP network. The left and right frames show the same data, however the right frame is plotted with a logarithmic y-axis. Each model is run 100 times and the average for each trial is plotted. The network has 3 hidden layers of 24 neurons each, a common PINN architecture. The batch size is $2^8 = 256$ samples.

We run extensive experiments to analyze the effect of varying batch size, network width, network depth, and number of derivatives. The results of the forward passes are summarized in Figure 3.8, and the results of the combined forward and backward pass are summarized in Figure 3.9.

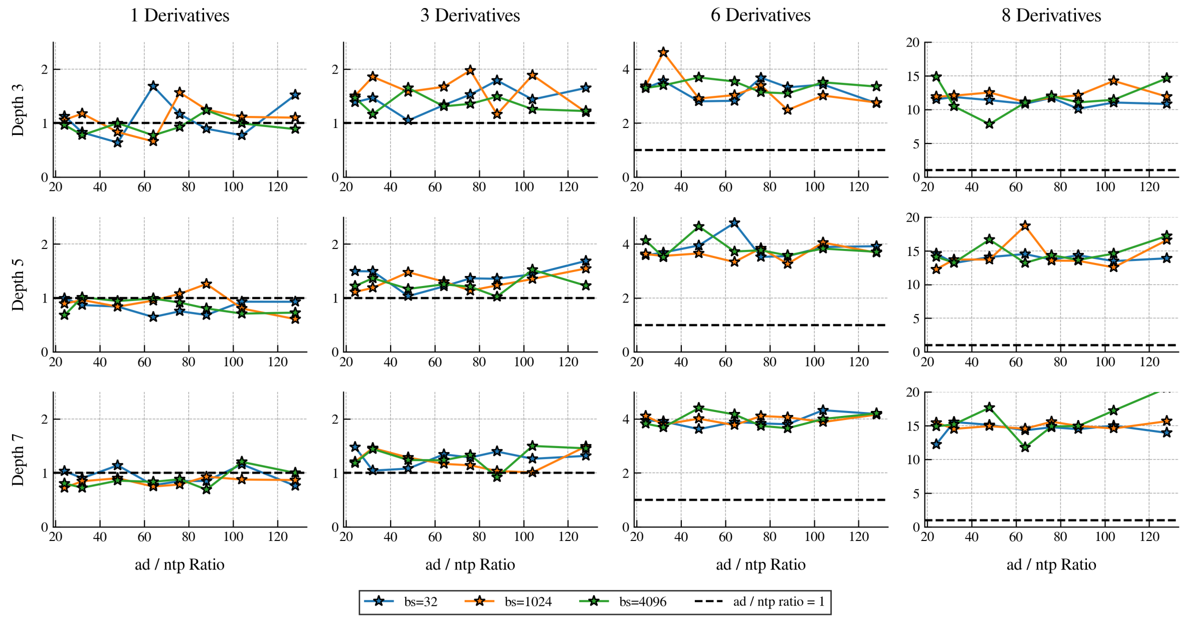


FIGURE 3.8. The ratio of forward pass run times between autodifferentiation and n-TP for a variety of network architectures, input batch sizes, and number of derivatives. A ratio greater than 1 indicates that n-TP was faster than autodifferentiation. The baseline ratio of 1 is plotted as a horizontal dashed line. All plotted data points represent the average of 100 trials.

We point out several salient features in these results. First, we observe a performance gap between n-TP and autodifferentiation for low derivatives. This is likely a consequence of implementation details, rather than a deficiency with the proposed methodology. The PyTorch implementation of autodifferentiation is heavily optimized for execution on a GPU, and while our implementation makes attempts at closing this performance gap, it is written in Python, rather than a lower level language such as `C++`, and lacks sophisticated optimization strategies. We suspect that a more refined implementation would close the gap seen for low derivatives.

Second, we observe that the performance gains afforded by n-TP decrease as we increase the batch size. We hypothesize that this effect is also due to a lack of optimization to take full advantage of the parallelized computational ability of the GPU in our implementation. For example, our implementation does not fully vectorize the computation of Equation (3.3) and thus does not take full advantage of the hardware scaling afforded by our GPU. Similarly, we observe that the performance gains from n-TP decrease as we increase the network width. We suspect that this is also a consequence of the lack of vectorization. Increasing either width or batch size scales the compute horizontally, and we have not fully optimized our implementation to account for this horizontally scaled compute.

Third, we observe that for all of the tested derivatives and batch-sizes, the standard PINN architecture of three hidden layers and 24 neuron widths [**84**] performs better with n-TP than autodifferentiation, at least for derivatives of order three or higher. This suggests that for PINN problems involving higher-order derivatives, n-TP can be used as a drop in replacement without any further implementation tuning (See Section 3.4.3 below).

Finally, we observe an apparent asymptote for the combined forward-backward pass times as the number of parameters and batch size increase (see the bottom rows of Figure 3.9). We suspect that this is because as we increase the relative number of FLOPs the theoretical gains from n-TP begin to dominate the superior optimization in PyTorch's autodifferentiation implementation. We hypothesize that with a stronger n-TP implementation we would begin to see similar asymptotic behaviors even when taking fewer derivatives.

We add that we could not compute more than nine derivatives using autodifferentiation because the required memory exceeded the 49 GB of memory available on our GPU. However, as further demonstration of the efficacy of n-TP we were able to take nine or more derivatives using our method while remaining on the single GPU.

3.4.3. End-to-End PINN Training.

Forward-Backward pass times should correlate to end-to-end model training but it is still important to measure the effect of the proposed modifications over the long time-horizons and multiple optimizers present in end-to-end PINN training. For example, our proposed n-TP method uses a different memory footprint for forward pass than autodifferentiation does. It can be difficult to reason theoretically about the effect that such changes will have to the end-to-end performance of machine learning models, and as such, empirical analysis is imperative to rule out performance degradation which arises as a consequence of the complicated end-to-end training testbed.

We find that the widespread use of the L-BFGS optimizer adds to the improvements afforded by n-TP. L-BFGS performs quasi-second order optimization using a line-search [**76**] which requires performing multiple forward passes through the network but only a single backwards pass. Thus, the forward pass performance seen in Figure 3.8 is expected to dominate and we expect that our unoptimized n-TP algorithm will outperform standard PINN implementations using autodifferentiation.

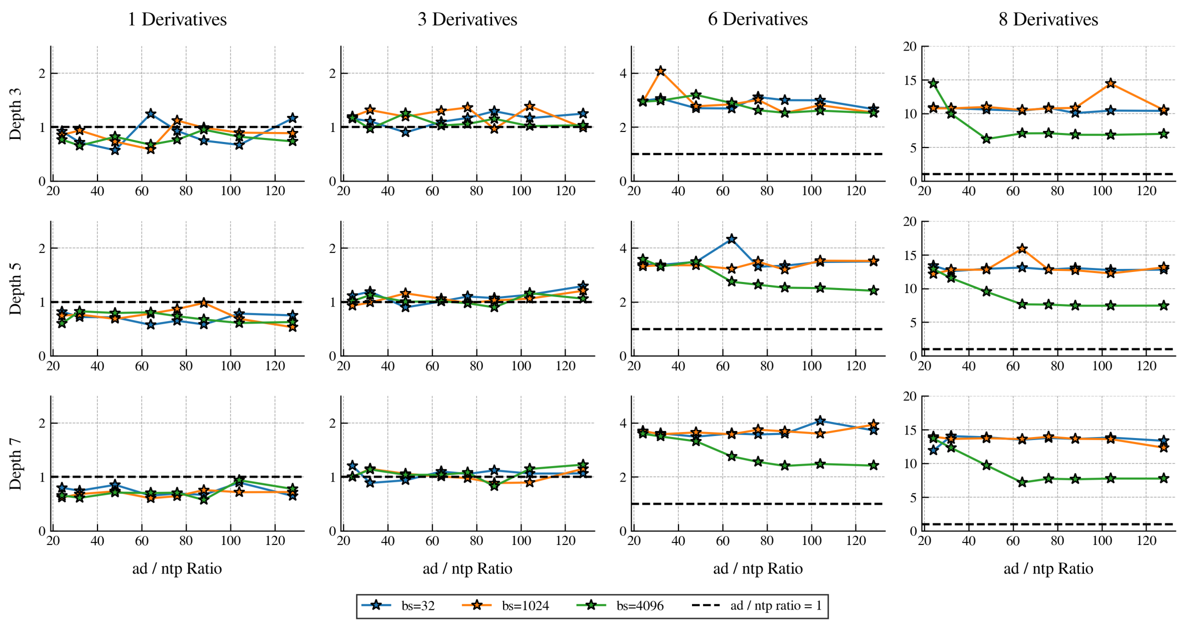


FIGURE 3.9. The ratio of combined forward-backward pass run times between autodifferentiation and n-TP for a variety of network architectures, input batch sizes, and number of derivatives. A ratio greater than 1 indicates that n-TP was faster than autodifferentiation. The baseline ratio of 1 is plotted as a horizontal dashed line. All plotted data points represent the average of 100 trials. The forward pass time ratio alone is plotted in Figure 3.8.

3.4.3.1. *Unstable Self-Similar Burgers Profiles.* In Chapter 2.1.4.3 we considered some properties of the self-similar Burgers equation, which we used as the asymptotic limit of the fractal Burgers dynamics. We recall the salient facts here once again.

Burgers equation is the canonical model for 1D shock formation phenomenon [47] and is given by the PDE

$$\partial_t u + u\partial_x u = 0. \tag{3.1}$$

Due to the nonlinear steepening of the wave profiles leading ultimately to a gradient blowup, studying the behavior of shock solutions near the singularity is difficult. Work in the 20th century explored the use of self-similarity to study the breakdown of smooth solutions near a shock [5]. Under the self-similar coordinate transformation

$$u(x,t) = (1-t)^{\lambda} U(x(1-t)^{-1-\lambda}), \quad X = x(1-t)^{-1-\lambda},$$

the PDE (3.1) becomes the ODE [39] for U in X

$$-\lambda U + ((1+\lambda)X + U)\, U' = 0, \tag{3.2}$$

for a scalar valued parameter $\lambda \in \boldsymbol{R}^{>0}$. While the solution of this particular problem is elementary and given implicitly by

$$X = -U - CU^{1+\frac{1}{\lambda}}, \tag{3.3}$$

the techniques used for the numerical analysis and solution of this problem can be applied to more challenging problems to yield highly non-trivial results about shock formation in complicated nonlinear equations [105].

From (3.3) we observe that the solution U will be smooth whenever $1+\frac{1}{\lambda}$ is an integer, and be physically realizable (odd) whenever $1+\frac{1}{\lambda} = 2k$ for some positive integer k [39]. Thus, the possible values of λ corresponding to smooth solutions are $\lambda = \frac{1}{2k}$ for $k = 1, 2, \cdots$. For all other values of λ the solution U will suffer a discontinuity at the origin in one of it's higher-order derivatives.

Our goal for this problem is to find these physically realizable solutions, a problem which is complicated by the fact that the profiles corresponding to $k = 2, 3, \cdots$ are physically and numerically unstable [39]. Traditional solvers will not converge to these solutions, and they do not manifest as real-world shocks due to a collapse towards the solution corresponding to $k = 1$. Regardless, these unstable profiles are important to understand mathematically and can give insights into the underlying behavior of certain systems. See for example the papers [22, 78], which apply self-similar methodology to perturbations of the self-similar Burgers equation 3.2.

Wang et al. [105] propose a methodology for solving Equation 3.2 for an unknown value of λ using PINNs to perform a combined forward-inverse procedure and simultaneously solve for U and λ. Such a methodology demonstrates the advantage that PINNs have in certain numerical settings, since solving this problem using traditional solvers is challenging. We refer the reader to the study by Biasi [8] which addresses a similar problem using traditional numerical methods and highlights the attendant difficulties.

3.4.4. Using n-TP for Higher-Order Profiles of Self-Similar Equations. The primary observation in [105] is that a solution to (3.2) is smooth for all values of λ, except at the origin, where a discontinuity will appear for derivatives $n \geqslant 1 + \frac{1}{\lambda}$. Thus, if we restrict the value of λ to $[1/3, 1]$ and enforce a smoothness condition on the third derivative of our neural network, we will converge to the unique smooth profile (if one exists) in the range $[1/3, 1]$. This is because for any non-smooth profile in this range, the third derivative or lower **must** be non-smooth at the origin. To find higher-order smooth profiles we can look between $[1/(2k+3), 1/(2k+1)]$ and enforce a smoothness condition on the $2k+3$-th derivative.

Using a PINN, we can enforce differentiability at the origin by taking sufficiently many derivatives there, since the neural network solution is smooth. This forces the solution to be smooth, which in turn gives gradient signal to push λ towards a valid value. We loosely follow the training schedule used in [105] to compute the first, second, third, and fourth profiles in quasilinear, rather than exponential time. The authors in [105] only computed the first and second profiles, so our results represent the first time the third and fourth profiles have been computed using PINNs. We stress that computing these solutions requires taking many derivatives and thus the n-TP formalism is well-suited for this type of problem. Additionally, computing the third or higher profile is extremely computationally intensive.

We note in passing that we were unable to reproduce the accuracy results claimed by the Wang et al. paper [105] and the authors do not provide access to their code. Regardless, our work is orthogonal to theirs and any future attempts at a reproduction of their work will benefit from using n-TP (see Chatper 3.4.5 for a further discussion).

We run our self-similar Burgers experiments using 64-bit floating point precision. We give a detailed description of our methodology and results below in Chapter 3.4.5, which we hope will contribute to the reproducibility of the [105] results.

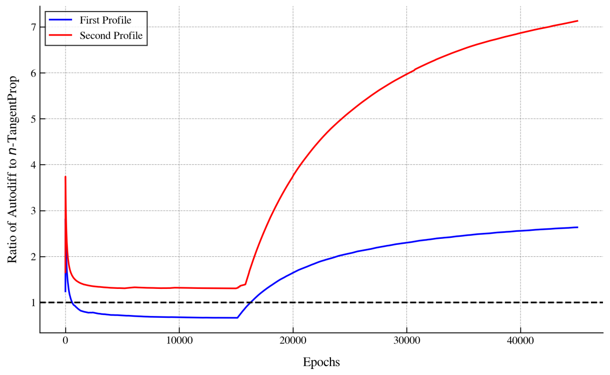


FIGURE 3.10. The ratio in runtime between autodifferentiation and n-TP as a function of number of epochs. The horizontal line in the bottom panel indicates a runtime ratio of 1. This shows that n-TP is 2.5x faster for end-to-end training than autodifferentiation for the first profile. We can also see that the end-to-end training for finding the second profile is 7x faster when using n-TP instead of autodifferentiation.

Figure 3.10 plots the ratio of execution times of autodifferentiation over n-TP and shows that we obtain significant speedups in computation time using n-TP instead of autodifferentiation. We were only able to compute the timing comparison between autodifferentiation and n-TP for the first two profiles since the computational time for the third profile using autodifferentiation exceeded our allowable computation time of 24 hours. For the first profile, which requires taking three derivatives, we obtain an end-to-end speed up of over 2.5 times. For the second profile, which requires taking five derivatives, we obtain an end-to-end speed up of over 7 times. We were able to compute the third profile, which requires taking seven derivatives, in a

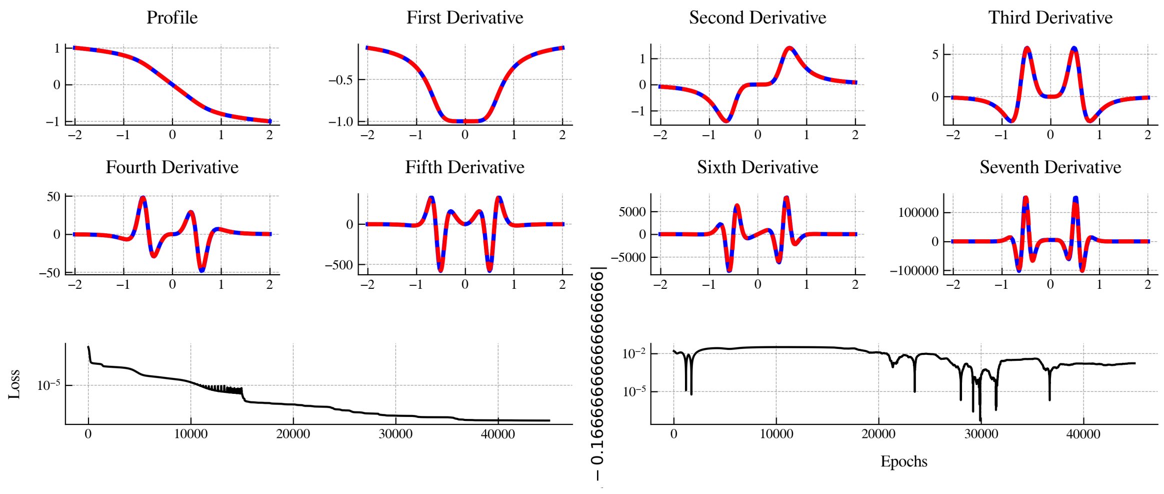


FIGURE 3.11. Results from training a PINN to solve (3.2) with λ constrained to the range $[1/7, 1/5]$. The only smooth solution to (3.2) contained in this parameter range corresponds to $\lambda = 1/6$. The first two rows show our learned solution (dashed red) and its derivatives compared to the true solution (solid blue). The last row (left) shows the PINN training loss as a function of epochs. The model was trained for 15k epochs using the Adam optimizer and 30k epochs using the L-BFGS optimizer. The last row (right) shows the inferred value for the parameter λ as a function of epochs. The bottom row is plotted with a logarithmic y-axis.

little under 1 hour using n-TP, and the projected time for auto-differentiation was over 25 hours, giving an expected speed-up of at least 25 times.

Using n-TP we were also able to compute the fourth profile, which requires taking nine derivatives. Using n-TP we were able to run the 45k epochs in a little under an hour and a half. We discuss our results further in Chapter 3.4.5, which we think are interesting in their own right. We stress that computing this fourth profile is untenable using autodifferentiation, as the time and space complexity render attempts at computation overwhelmingly challenging on conventional hardware. We estimate that computing the fourth profile using autodifferentiation would take at least 100 hours (about four days) if one had a GPU with sufficient memory. Thus, we expect that we would get a speedup of at least 65 times.

Observe from Figure 3.10 that the most dramatic improvements come when we switch to the L-BFGS optimizer, which uses multiple forward passes to perform a line search. Because n-TP has more favorable forward pass dynamics (c.f. Figures 3.6 and 3.7), the performance improvements are much more pronounced during L-BFGS optimization. This emphasizes the advantage afforded by n-TP: to obtain high-accuracy results we often use L-BFGS and Sobolev loss (see Equation 3.2). These two accuracy improvements require taking higher-order derivatives more frequently, which are the two areas that n-TP shows the best improvement in. Thus, for the high-accuracy training phase for PINNs, n-TP yields significant performance improvements.

We suspect that the dip below a ratio of 1 that we see in Figure 3.10 for computing the first profile can be mitigated through further optimizations of our implementation, and we hypothesize once again that the dip is likely due to efficiencies afforded by graph pruning and operator fusing in PyTorch.

Figure 3.11 shows the result of training a PINN to find the third smooth profile of (3.2). This is the first time that we are aware of that this profile has been numerically computed with a floating value of λ using either a PINN or a traditional solver. We show the learned solution with a dashed red line superimposed over the true solution in blue. This profile is already computationally expensive to compute using autodifferentation and n-TP opens the door to performing novel studies on equations requiring a high-number of derivatives.

3.4.5. Additional Details for the Self-Similar Burgers Experiments. We report the results from running the self-similar Burgers experiment to find the smooth stable and unstable profiles. We were not able to reproduce the accuracy reported in [**105**], however we think that our results are important in demonstrating that their proposed methodology is robust, at least in theory. Furthermore, we report several new observations that we think are relevant to the training dynamics for such a problem.

We train our network using a Sobolev loss function (see Equation 3.2 function with $m = 1$ and additionally add a high-order loss term

$$L^*(u_\theta) = \frac{1}{N^*} \sum_{k=1}^{N^*} |\partial_x^n R(u_\theta, x_k)|^2,$$

where R is the residual of the self-similar Burgers equation and the samples x_k are taken from a small subset of collocation points centered at the origin, not the entire training domain. Our implementation contains many more subtle details that we omit for the sake of brevity and we encourage the reader to download our code to see the full implementation.

While we were not able to match the accuracy reported by [**105**], we were able to get our implementation to perform well in finding the first three smooth solutions to (3.2). However our method did not satisfactorily solve for the fourth profile. We discuss this in more detail below.

For all of the profiles we found, we report our inferred value of λ as a function of training epochs. We think that this metric is an important scalar quantity and it's evolution is not reported in the original paper [**105**]. Of particular importance is the apparent inability of the Adam optimizer to satisfactorily converge to the correct value of λ. We see a sharp decrease in the error of λ once we begin to use the

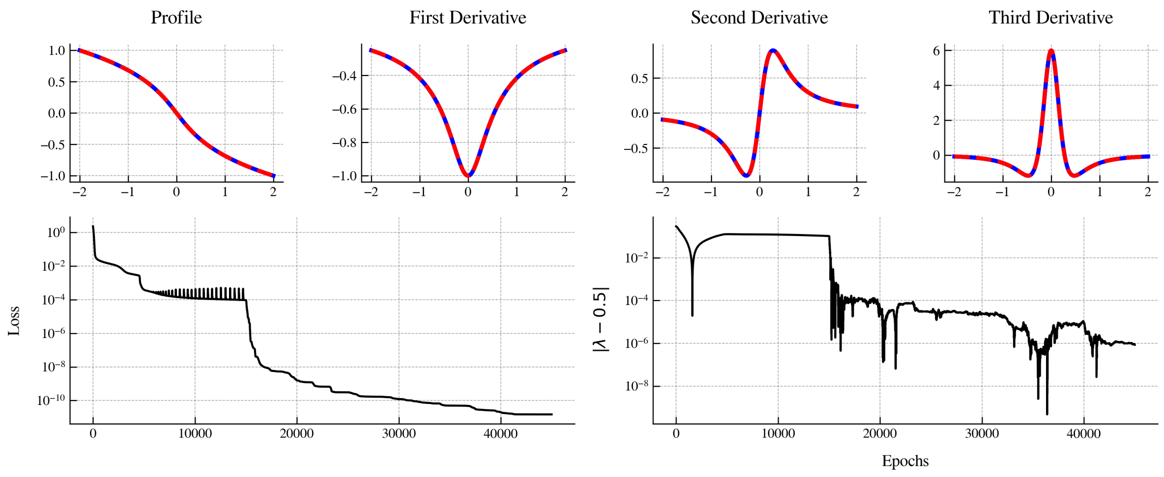


FIGURE 3.12. Results from training a PINN to solve (3.2) with λ constrained to the range $[1/3, 1]$. The only smooth solution to (3.2) contained in this parameter range corresponds to $\lambda = 1/2$. The first row shows our learned solution (dashed red) and its derivatives compared to the true solution (solid blue). The second row (left) shows the PINN training loss as a function of epochs. The model was trained for 15k epochs using the Adam optimizer and 30k epochs using the L-BFGS optimizer. The second row (right) shows the inferred value for the parameter λ as a function of epochs. The bottom row is plotted with a logarithmic y-axis.

L-BFGS optimizer. We think that this phenomenon is interesting and may indicate that the first order derivatives of the residual with respect to the parameter λ is insufficient to capture the true dependency of the solution on λ. Understanding this dependency more deeply may lead to better training algorithms for these types of problems.

Notably, our code failed to adequately converge to the fourth profile corresponding to $\lambda = \frac{1}{8}$ (see Figure 3.14). Due to the nature of this work we did not pursue this point further and want to emphasize that we are not claiming that the methodology proposed in [**105**] cannot be applied to higher-order profiles. We hypothesize that the nature of the problem makes it more difficult for PINN or non-PINN solvers to find a solution. We are constraining the ninth derivative to be close to zero near the origin, but due to the relatively large magnitude of the ninth derivative, minor fluctuations in the network output will result in large changes to the ninth derivative. This alone may be enough to render our solver incapable of converging to the desired solution as we are using a fixed ratio to balance the relative terms in our loss function (see [**11**] for further discussion about why multi-target training is difficult). Put another way, we suspect that the loss function we are using does not properly account for the fact that we are taking nine derivatives and that as a consequence the parameter λ is not receiving good gradient signal.

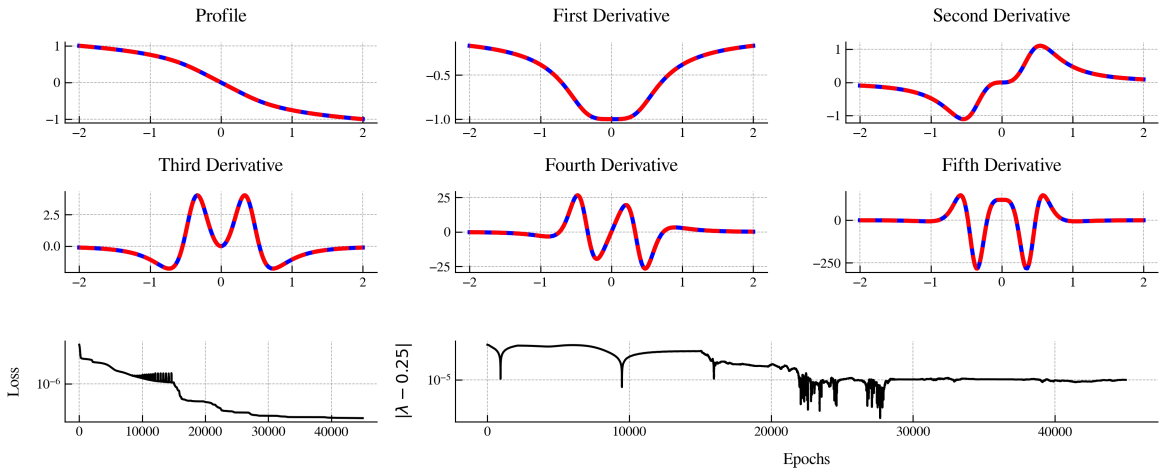


FIGURE 3.13. Results from training a PINN to solve (3.2) with λ constrained to the range $[1/5, 1/3]$. The only smooth solution to (3.2) contained in this parameter range corresponds to $\lambda = 1/4$. The first two rows show our learned solution (dashed red) and its derivatives compared to the true solution (solid blue). The last row (left) shows the PINN training loss as a function of epochs. The model was trained for 15k epochs using the Adam optimizer and 30k epochs using the L-BFGS optimizer. The bottom row (right) shows the inferred value for the parameter λ as a function of epochs. The bottom row is plotted with a logarithmic y-axis.

The purpose of this study was not to find the optimal hyperparameters for computing these higher-order profiles. Rather, the point of our study is to demonstrate that computing these higher-order profiles is feasible. We stress that using autodifferentitation to find this fourth profile would likely take several days on a state-of-the-art GPU, and we were able to compute it in less than two hours. We leave the refinement of model accuracy to future studies.

3.5. Conclusions

We introduced the n-TP formalism and demonstrated both theoretically and empirically that implementing our formalism in the context of PINNs dramatically reduces training times. We showed that for derivative-intensive PINN applications like finding high-order solutions to self-similar equations, n-TP not only offers improvements in end-to-end training times but allows the computation of previously untenable solutions. Our results are a step in the direction of making PINNs a more competitive numerical method for difficult forward and inverse problems. Because our methodology is exact, any existing PINN study will benefit from using n-TP. For this reason we recommend that our formalism be adopted by PINN implementations going forward to ensure faster training of PINNs.

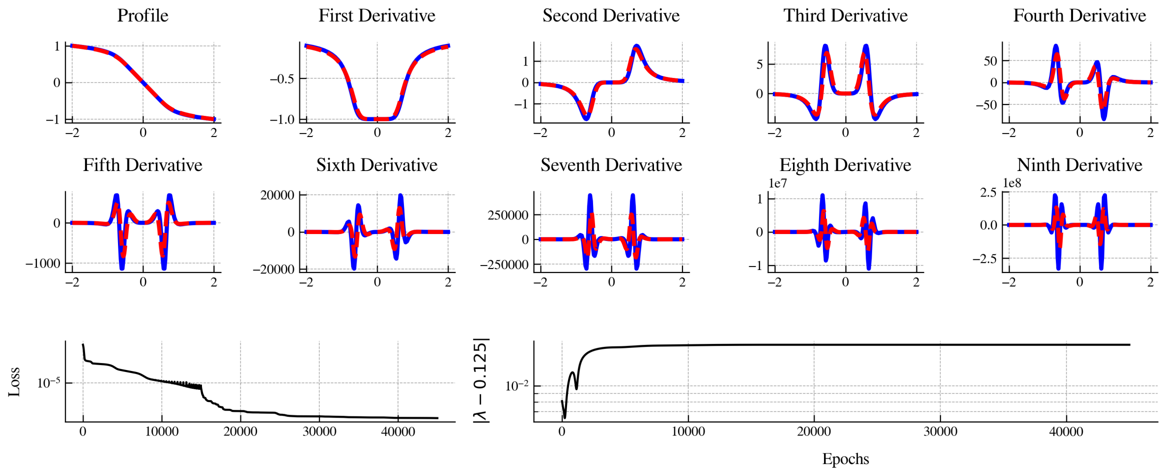


FIGURE 3.14. Results from training a PINN to solve (3.2) with λ constrained to the range $[1/9, 1/7]$. The only smooth solution to (3.2) contained in this parameter range corresponds to $\lambda = 1/8$. The first two rows show our learned solution (dashed red) and its derivatives compared to the true solution (solid blue). The second row (left) shows the PINN training loss as a function of epochs. The model was trained for 15k epochs using the Adam optimizer and 30k epochs using the L-BFGS optimizer. The second row (right) shows the inferred value for the parameter λ as a function of epochs. The bottom two rows are plotted with a logarithmic y-axis.

We hope that our work allows the PINN community to explore more complicated problems, deeper and wider network architectures, and allow for researchers who do not have access to powerful computer to participate in furthering PINN research.

In this paper we have not focused on the optimization of our algorithm. We think that with optimization choices like implementing the underlying logic in `C++` instead of Python that the performance gap between n-TP and autodifferentiation would widen even further.

CHAPTER 4

Statistical Efficiency: QLIP & Enhanced Vision Priors

This final chapter is based on the author's previous work [21]. In this chapter we will leverage the understanding gleaned from working with PINNs to apply the same high-level philosophy to a problem in a completely different domain: multi-modal large language models. While the learning target we use for this chapter indeed satisfies the criteria for an overfitting MLP problem as discussed in Chapter 1, the size of the networks under consideration complicates things, as we discuss below. However, the intuitions and implications discussed in the Introduction inform our choices in for the MLP training in the present Chapter.

4.1. Introduction

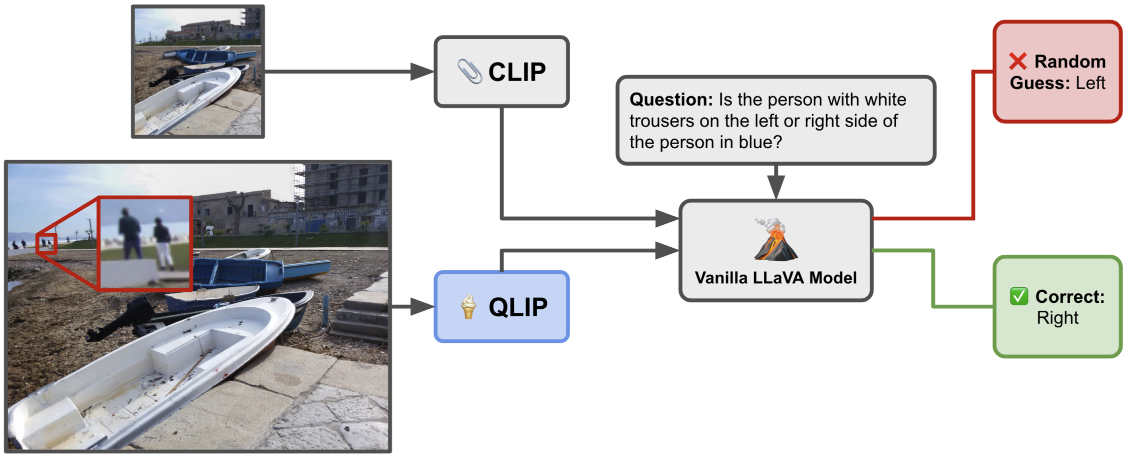

CLIP
QLIP
Question: Is the person with white trousers on the left or right side of the person in blue?
Vanilla LLaVA Model
Random Guess: Left
Correct: Right

FIGURE 4.1. QLIP is a **drop-in replacement** for CLIP which allows models like LLaVA to perform inference on arbitrarily large images. In our experiments we find that vanilla LLaVA + QLIP gives **+13.6%** accuracy on the challenging V^* benchmark **with no re-training or fine-tuning**. The example in the figure above demonstrates an instance where CLIP cannot correctly get the answer because (a) in the cropped version of the image the person in question is not present, and (b) if we use a padded image the person will be too small to provide meaningful signal to model.

Multimodal Large Language Models (MLLMs) have shown impressive multi-modal question answering ability, yet recent work has highlighted a deficiency whereby these models struggle to answer questions about fine-grained visual details [89, 108]. MLLMs like the popular LLaVA family [64, 65] contain a vision encoder and visual projector which embed visual information into a sequence of tokens within a shared visual-linguistic embedding space before passing these tokens to an LLM. Recent work has demonstrated that, for many visual question answering (VQA) tasks, model performance is nearly unaffected by the removal of a high number of visual input tokens [52, 61, 93]. It has also been shown that models like LLaVA overly rely on information from the vision encoder's `[CLS]` token [116] to answer questions. Because the `[CLS]` token is a high-level encoding of an image's content, reliance on this token does not aid in fine-grained visual analysis.

We posit that this failure to perform satisfactorily on fine-grained VQA tasks is neither a deficiency in the training process of the MLLM nor a deficiency in the representations which can be encoded by the vision encoder. Prior works aimed at modifying the vision encoder or projector architectures have implicitly assumed that the failure mode is caused by CLIP itself, but this is only partially true. Li et al. [61] show that the vanilla LLaVA architecture with the CLIP encoder is capable of much better VQA performance, but requires the "correct" tokens to be fed to the language model. Similarly, Li et al. [62] show that the information from the CLIP encoder is often sufficient for certain vision tasks or VQA, however, the models often do not adequately use the given information.

We argue that the failures incurred while using the CLIP encoder can be attributed to two specific biases induced by the inductive priors which were implicitly assumed during the training of the CLIP encoder.**Mesoscopic Bias** occurs because CLIP uses a uniform grid-patchification (UGP) strategy [34, 82] and manifests as the model implicitly treating uniform grid cells at a specific image scale as the fundamental unit of semantic meaning. **Interpolation Bias** arises as a consequence of CLIP being trained with fixed positional embeddings on fixed-resolution images and prevents CLIP from handling high-resolution images.

Previous work has focused on training a new vision encoder to replace CLIP [46, 64, 68, 89], but this requires re-training the entire MLLM, which is expensive and often not feasible for existing MLLM users. In this work, we take a minimally invasive approach and carefully reason through the consequences of updated vision priors. This lead us to a *light-weight, content-aware, drop-in* modification to the CLIP encoder which we name QLIP.

QLIP empowers CLIP based MLLMs to automatically process arbitrary resolution input images, while adaptively scaling the number of input tokens based on the semantic content of the image. We find that reducing the number of input tokens has beneficial effects beyond reducing computation. We find that reducing image tokens in a reasoned way can reduce model hallucination and improve fine-grained VQA. To assess both the effectiveness and efficiency of QLIP, we apply it to the LLaVA-1.5 family of MLLMs for VQA. We emphasize fine-grained visual tasks like the challenging V^* benchmark [108]. Our method achieves a 13.6% improvement on V^*, reduces hallucination rates as measured by the POPE F1 score [63] by 5.2, and yields notable improvements across other multi-modal benchmarks including MME [43] and RealWorld-QA [110].

We accomplish this using two novel strategies. First, to address the mesoscopic bias we introduce a non-uniform patchification scheme based on image quadtrees [53]. Our patchification procedure is *adaptive, tunable, and training-free*, and implicitly treats semantically similar regions of the image as the fundamental unit of semantic meaning instead of UGP. Second, to address the interpolation bias, we train a small MLP network to interpolate the fixed positional CLIP embeddings while maintaining usable positional signals for downstream models. This small interpolation network requires little training yet is both highly effective and generalizable.

In this chapter we do the following:

(1) We identify two fundamental biases in the CLIP vision encoder, i.e. mesoscopic bias and interpolation bia, and propose quantitative measures of both.

(2) We introduce QLIP, a lightweight, drop-in modification for CLIP that supports arbitrary image resolutions and adaptively scales the number of the image tokens based on image content. QLIP directly mitigates the aforementioned biases without modifying the original encoder weights or requiring expensive re-training of the MLLM.

(3) We empirically validate the effectiveness of QLIP by integrating it into the LLaVA model family [64, 65] and demonstrating substantial performance improvements. Our results are achieved without any supervised fine-tuning or re-training of the model backbone. For the challenging V^* benchmark, we achieve a significant improvement of +13.6% accuracy using LLaVA 13B with QLIP, outperforming the previous SoTA CLIP-based LLaVA results by +3.1% [89].

4.2. Why CLIP Fails at Higher Resolutions

The CLIP vision encoder is trained at a fixed input resolution using learned absolute positional encodings [82]. This design introduces two notable and consequential biases. First, because the positional encodings are absolute rather than relative, they do not generalize beyond the spatial grid of the training resolution. This limitation constrains the encoder's ability to handle images at arbitrary resolutions. Second, the encoder is trained exclusively on fixed-scale images, which biases the encoder towards only recognizing features at a specific mesoscopic spatial scale. For example, consider the elephants in Figure 4.2. The CLIP encoder is most likely to understand the middle (meso) image as containing an elephant, rather than the left or right images. This is because during training it is unlikely that the leftmost image would be labeled as having an elephant in it and the rightmost image may be too zoomed in to distinguish from other concepts.

To accommodate inputs of varying resolutions, the standard CLIP preprocessing pipeline resizes and crops raw images to a fixed square shape. This process inherently biases the model toward a particular mesoscopic image scale (See Figure 4.2). We refer to the former limitation as **interpolation bias** and the latter as **mesoscopic bias**.

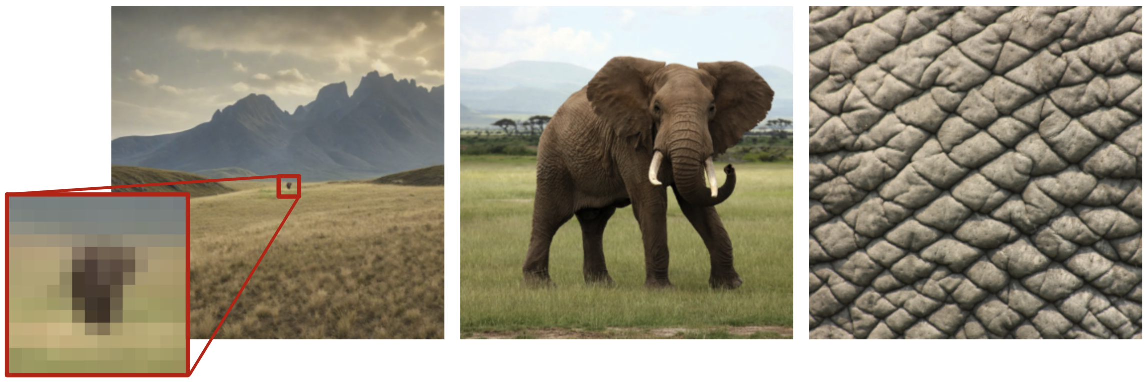


FIGURE 4.2. An example of the same semantic feature (`animal:elephant`) at three different spatial scales. These photos could be accompanied by the question `What animal is shown in this photo?` For the leftmost image the elephant fits into a single patch. Without memorization it is unlikely for any classifier to be able to accurately identify the pixilated blob as an elephant instead of, for example, a horse or a buffalo.

Quantification of Interpolation Bias Consider a single image $\mathcal{I}$ rendered at two different resolutions, $R_1 = (H_1, W_1)$ and $R_2 = (H_2, W_2)$. We denote the corresponding resized images as $\mathcal{I}_{R_1}$ and $\mathcal{I}_{R_2}$, respectively. Since both images originate from the same source and contain identical semantic content, one would reasonably expect the CLIP `[CLS]` token embedding to remain invariant or at least approximately invariant across nearby resolutions. Under this assumption, the cosine similarity between the corresponding CLIP embeddings, $\mathcal{E}1 = \text{CLIP}(\mathcal{I}R_1)$ and $\mathcal{E}2 = \text{CLIP}(\mathcal{I}R_2)$, serves as a measure of the deviation introduced by resolution changes. To quantify the extent to which the positional embeddings contribute to this deviation, we define the interpolation bias as:

$$\mathcal{B}_{\text{Interp}}(\mathcal{I}) := ||\, \nabla_{\mathcal{P}} \text{CS}(\mathcal{E}_1, \mathcal{E}_2)\, ||_2\,, \tag{4.1}$$

where $\mathcal{P}$ denotes the additive positional encodings applied to patch embeddings during the CLIP encoding process [82]. Thus, the interpolation bias is the norm of the derivative of the cosine similarity between resolutions, with respect to the positional encodings.

We have assumed that the cosine similarity between the two resolutions should be similar, but this isn't even necessary for us to effectively measure interpolation bias. We can consider instead two images which have all of their pixels set to the same color. Thus, the patch embeddings are identical, as is the semantic content of the image (although image scale based information may still be propagated). In this case we really do expect that the embeddings are the same, since the images are the same. In this case we may plausibly expect that the cosine similarity itself is a measure of interpolation bias. However, the definition given in 4.1 is preferable because it allows us to isolate the contributions to the difference between image resolutions purely accounted for by the positional embeddings. Finally, we note that recent work has shown that MLLMs rely heavily on `[CLS]` tokens, making them a relevant quanitity to study in practice [116].

Quantification of Mesoscopic Bias The mesoscopic bias is easier to quantify because we can simply remove the positional encodings and look at the cosine similarity of the `[CLS]` token embeddings at different image sizes. To this end, consider an image $\mathcal{I}$ with resolution $N \times N$ and then consider the same image rescaled to 336×336, which we denote $\mathcal{I}_{336}$. Let $\mathcal{E}^z = \text{CLIP}^z(\mathcal{I})$, $\mathcal{E}^z_{336} = \text{CLIP}^z(\mathcal{I}_{336})$ be the respective `[CLS]` embeddings after setting the positional encodings to zero. Then

$$C^z_{N\to 336} := \text{CS}(\mathcal{E}^z, \mathcal{E}^z_{336})$$

captures the degree to which the overall embedding has changed as an effect of the mesoscopic scale of the input images.

Again, we intuitively expect that the `[CLS]` token captures semantic information about the image, and that for rescaling a given image, the `[CLS]` token should remain approximately constant since the content of the image is not changing.

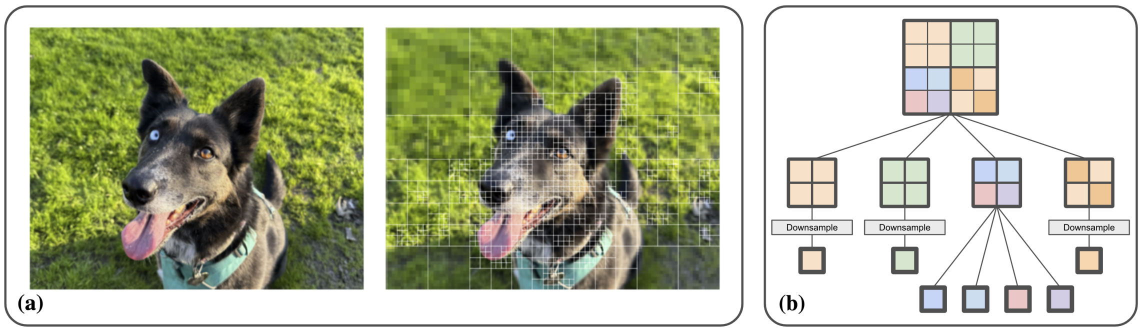


Figure 4.3. **(a)** An example of the quadtree patchification (QtP) applied to a high-resolution image. QtP uses only 25% of the original number of tokens yet retains a high-degree of semantic information. Photo courtesy of first author. **(b)** A schematic of a 4×4 patch image being decomposed into 7 leaf patches using a quadtree. Leaves which consist of more than a single patch are downsampled to the patch size.

4.3. Addressing the Mesoscopic and Interpolation Biases

We identify and address two implicit inductive priors underlying the CLIP encoder, noting that these assumptions were likely adopted primarily for engineering practicality.

The first prior is that UGP represents the fundamental unit of semantics which we address by replacing UGP with a *content-aware quadtree patchification* (QtP). The second prior is that images can be effectively represented by center-cropping and rescaled to a fixed resolution which we address by training a small interpolation network.

4.3.1. Vision Quadtree Mitigates the Effects of Mesoscopic Bias. Natural images do not contain uniformly distributed information throughout their sub-images. In general, semantic information can continue to be extracted even when large portions of the image are subjected to extreme levels of information degradation at the pixel level (see Figure 4.3). This is the reason that compression algorithms like JPEG work [**99**].

We derive a strategy for adaptively merging adjacent patches in an attempt to increase the quality of the visual signal coming from the vision encoder. This strategy is based on the intuition that many pixels in a given image do not contribute to the representation of the semantic content of the image. We propose using a quadtree [**53**] structure to adaptively select tokens based on some property intrinsic to the sub-images themselves. Quadtrees, as applied in image processing, are hierarchical image representation trees which generalize a binary tree into two dimensions. At the root of the tree is the original image, and at each level we subdivide the image into four , until we reach the leaf nodes which represent patches (see Figure 4.3, **(b)**). We can then prune the tree according to some selection rule and the resulting leaf-nodes consist of all sub-images which satisfy some maximal condition. We apply downsampling to the leaf-nodes which are larger than the CLIP encoder's patch size to obtain a sequence of patches that can be fed to the CLIP vision encoder. In theory semantically irrelevant portions of the image are downsampled back to the mesoscopic scale that CLIP expects, and important tokens which represent a small portion of the visual field are effectively upsampled into the same scale (see Figure 4.3, **(a)**).

In what follows, we use the L^∞ sub-image averaged gradient as the selection criteria. In particular, an image I is a leaf-node if it is the patch-size or otherwise if

$$\mathcal{D}(I) := \max_{x,y}(\, \partial_x I + \partial_y I \,) < \alpha, \tag{4.1}$$

where α is a pre-chosen selection constant. We also test a random selection strategy as an ablation for our selection strategy. More details are contained in Chapter 4.3.4.1.

4.3.2. Coordinate-Based MLP Mitigates the Effects of Interpolation Bias. The CLIP vision encoder consists of two mechanisms that work in concert to map information from the pixel space into the embedding space.

Let $\mathcal{P} = \{p_i\}_{i=1}^N$ be a set of patches with coordinates $\mathcal{X} = \{(x_i, y_i)\}_{i=1}^N$, $(x_i, y_i) \in [-1,1]^2$. The CLIP encoder can be understood as taking $\mathcal{P}$ and $\mathcal{X}$ and producing a sequence of tokens $\mathcal{S} = \{s_i := \boldsymbol{E}(p_i) + \boldsymbol{M}(x_i, y_i)\}_{i=1}^N$ along with a [CLS] token $\boldsymbol{E}_{\texttt{[CLS]}}(\mathcal{P})$. The [CLS] token is obtained by a pooling-type operation in the embedding dimension over $\mathcal{S}$.

CLIP is trained on 336×336 images decomposed into a series 14×14 patches using the standard UGP [34, 82]. There will then be $24 \times 24 = 576$ patches and to each of these patches CLIP associates a positional embedding $\mathcal{E}_{ij} \in \mathbf{R}^{1024}$, where $1 \leqslant i, j \leqslant 24$ respectively index the rows and columns of both $\mathcal{E}$ and the grid of patches. For this patchification we have $\boldsymbol{M}(-1 + \frac{2i}{23}, -1 + \frac{2j}{23}) = \mathcal{E}_{ij}$. We will extend $\boldsymbol{M}$ to the entire square $[-1,1]^2$ so that we can natively handle images of any resolution and apply our QtP. We choose to train an MLP which gives us a high-degree of expressivity and will be trained using our new inductive priors.

To do this, we assume that the interpolation bias (2.6) should be zero when we re-size an image. Directly training on (2.6) proves challenging in practice however, because differentiating the model with respect to the positional encodings prior to a backwards pass is already prohibitively expensive. This is becaus the CLIP neural network is several orders of magnitude larger than the PINNs we were working with in Chapters 3 and **??**. Thus, we seek some approximate analog for training our network.

To this end, we assume that the [CLS] token should remain invariant under small changes to the input image resolution. We generalize this assumption to broadly assume that the [CLS] token should remain invariant under any change to the input image resolution, understanding that this assumption does not fully account for effects from mesoscopic bias (see Section 4.2). Regardless, despite the error in this assumption we find that it serves as a suitable learning target.

Thus, if $\mathcal{G}$ is the standard UGP associated to the image I_{336} and $\mathcal{P}$ is a patchification associated to the image I_N, then we expect that

$$L_{\texttt{[CLS]}} := ||\, \boldsymbol{E}_{\texttt{[cls]}}(\mathcal{G}) - \boldsymbol{E}_{\texttt{[cls]}}(\mathcal{P}) \,||_{L^2} = \text{small}, \tag{4.2}$$

and this provides a target for training the MLP. However, in practice $L_{\texttt{[CLS]}}$ is insufficient for training since the pooling means that as long as $\sum_{ij} \mathcal{E}_{ij} = \sum_i \boldsymbol{M}(x_i, y_i)$, then the [CLS] embedding will be constant. Because we are attempting to train a drop-in modification for CLIP, we must make sure that downstream applications are minimally affected. In particular, since downstream MLLMs utilize the positional information from CLIP and were trained using CLIP's positional encodings, we must ensure that the MLP positional embeddings match the CLIP positional embeddings on the standard 24×24 grid. I.e. we need to interpolate the fixed positional encodings in such a way as to minimize the downstream consequences. Thus we additionally aim to minimize the residual L^1 error:[1]

$$\mathcal{R}(\boldsymbol{M}, \mathcal{E}) := \frac{1}{576} \sum_{i=1}^{24} \sum_{j=1}^{24} \left| \boldsymbol{M}\left(-1 + \frac{2i}{23}, -1 + \frac{2j}{23}\right) - \mathcal{E}_{ij} \right|. \tag{4.3}$$

Thus we arrive at a suitable loss function for the MLP training:

$$\text{Loss} = L_{\texttt{[CLS]}} + \gamma \mathcal{R}, \tag{4.4}$$

where γ is a hyperparameter to balance the relative effects of the two components of the loss. Training is stable and we include additional training details in Chapter 4.3.4.

We now highlight the close relationship between the present MLP training target and the overfitting neural networks discussed in Chapter 1. The loss term $\mathcal{R}$ is a direct interpolation term, and minimizing it ensures that QLIP agrees with CLIP on the default grid positions. However this alone is not useful, we must also make sure that QLIP performs well when evaluated at points which are not on the default grid. The $L_{\texttt{[CLS]}}$ term enforces the behavior away from the default grid. Ideally we would replace $L_{\texttt{[CLS]}}$ with direct optimization of (4.1), but as mentioned above, this proved prohibitively expensive.

This is an unfortunate drawback which necessarily decreases the efficiency of our MLP training, since the noise from training on natural images necessarily leaks into the gradient descent for the MLP. On the other hand, training over batches of images should bring the expectation of $\nabla_\theta L_{\texttt{[CLS]}}$ roughly in line with $\nabla_\theta \|\nabla_{\boldsymbol{P}} \text{CS}(\mathcal{E}_1, \mathcal{E}_2)\|_2$, since having zero interpolation bias would naturally minimize the loss term $\nabla_\theta L_{\texttt{[CLS]}}$ with respect to the model parameters (which only affect the positional embeddings).

Thus our present MLP training target puts the MLP into the overfitting regime with a noisy auxiliary constraint. This intuition is further justified by our hyperparameter tuning, which reveals that keeping the two loss terms in balance through setting $\gamma = 1$ yields the best results, indicating that the noise from the images is not too great to overwhelm the overfitting, and that the overfitting term $\mathcal{R}$ remains important for getting good results, see Chapter 4.3.4.

4.3.3. Training the Interpolation Network. We train the MLP for 100 epochs with the Adam optimizer [**55**] on the training split of the Imagenette dataset [**51**]. This dataset is a small subset of Imagenet [**32**] with only 10 classes, and consists of about 10k images. We argue that the choice of dataset does not matter much for the MLP training because the embedding function $\boldsymbol{M}$ is independent of the image content

[1]We found that L^1 loss was better than L^2 loss since we aim to get $\mathcal{R}$ to be smaller than 5×10^{-7}. See Chapter 4.3.4 for more details.

and the most important aspect of training is interpolating the fixed positional embeddings. We found that training on images was important for convergence though, and that training on pure colored images with random quadtree splits did not train quickly enough. We leave investigation of this phenomenon to future work.

Training took 11 hours on four NVIDIA L40S GPUs. We train with a batch size of 14, with images kept at their either their native resolution or smallest edge of length 560, whichever is smaller. We kept $\gamma = 1$. For our MLP architecture we use four hidden layers and pass the input features through a Fourier features layer [**94**] with 48 Fourier features. We chose to use Fourier features since preliminary analysis indicated that the default CLIP positional embedding function is high-frequency, and our analysis of the overfitting regime from Chapter 1 suggests that Fourier features will help training converge. Also following the findings from Chapter 1 we do not do any layer normalization and we fully sample the default embedding grid at each iteration. See Chapter 4.3.4 for a discussion about how we chose model hyperparameters.

4.3.4. Detailed Training.

4.3.4.1. *Detailed Implementation of a Quadtree.* To build a quadtree out of patches requires an image to be (a) square with (b) side lenghts consisting of 2^N patches for $N \in \mathbb{N}$. Obviously we are able to apply our methodology to images which are not of this size and we explain how we do so.

For concreteness, suppose we are given an image consisting of $M \times N$ patches. We can always center crop the images to the nearest patch size at a loss of at most 13 pixels. For this paper we first resize the smallest edge to our target size, then center-crop the image so that the longest side is also an integer number of patches.

Next, we find a grid of sub-images which maximally covers the original image, and where each of the sub-images in the grid is square with side lengths of 2^P for some P. For the remaining patches we leave them as is and pass their embeddings to the LLM. This process maximizes the number of patches in the image that can be subject to QtP. See Figure 4.4 for an example of this methodology applied to an image.

4.3.4.2. *Hyperparameters and Training Setup.* In our ad-hoc testing we quickly determined that for benchmark performance the MLP interpolation error was much more significant than the overall `[CLS]` embedding error. Therefore, our subsequent training experiments were targeted primarily at reducing MLP error.

We did not perform an extensive hyperparameter sweep over MLP architectures because the cost was prohibitive given our available compute. In what follows we describe our findings as we manually swept in individual directions to ablate our training hyperparameters.

- L^2 loss for `[CLS]` tokens is better than cosine similarity.
- L^1 loss for interpolation loss is better than L^2 loss. We found that the L^2 version of (4.3) was made smaller during training if we used the L^1 loss as the actual training target. We suspect that this is because the L^2 loss gets quite small (on the order of 10^{-5} to 10^{-7}, and stops sending meaningful signal to the weights.
- We swept four orders of magnitude for γ, $\gamma = 10^3, 10^2, 10, 1, 0.75$. We found that $\gamma = 1$ produced the best results and had the most stable training dynamics.
- Training on larger images produces better results but is more computationally expensive. The larger images seemed to give better results but took too long to perform meaningful sweeps over. We opted for a small batch size of 14 since it accelerated training while continuing to produce satisfactory results. We arrived at this number by choosing the maximal image size we were willing to train on and then saturating the GPUs.
- Dynamics appear stable regardless of batch size. We found that even with a very small batch size of 1 or 2 the training remained stable.
- Depth 4 MLP is better than a depth 2 MLP. We found that increasing the MLP depth from 2 to 4 gave better results and faster convergence of the interpolation error. We did not try depths greater than 4.

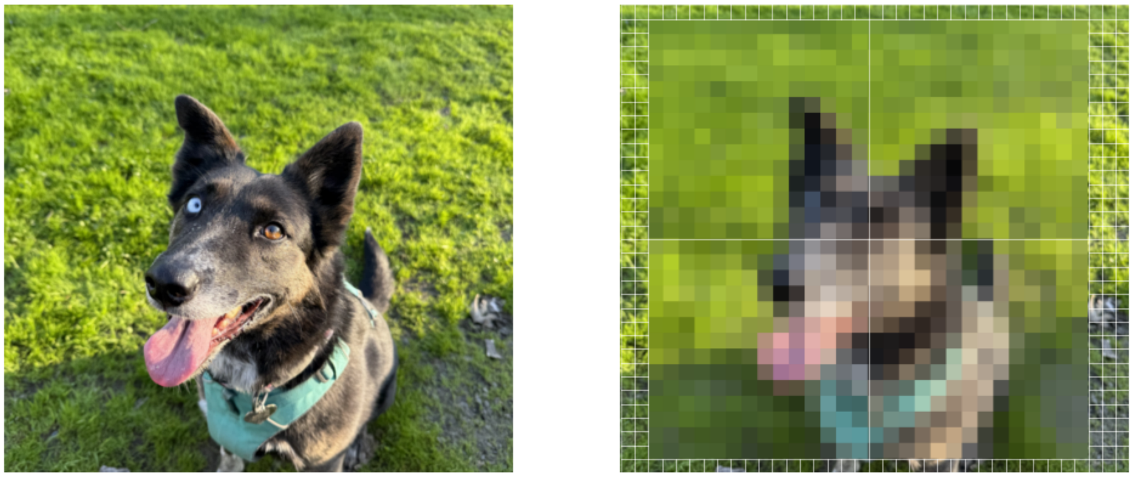


FIGURE 4.4. A quadtree applied to the image used in Figure 4.3, except with a different image size. The image in Figure 4.3 is 672×896, which can be decomposed into a 3×4 grid of 224×224 sub-images. Since $224 = 2^4 \times 14$ each of these 12 sub-images can have a QtP. The image in this figure is 476×518, which cannot be divided into QtP sub-images. The maximal grid of QtP-enabled sub-images is the 2×2 grid of 224×224 sub-images which are outlined in this Figure. Note the remaining patches are left as is around the border of the image.

- Fourier features: We tried 16, 32, and 48 Fourier features and found that 48 yielded the best results.
- We used a cosine learning rate scheduler and did not experiment with adjusting the schedule. See our code for full details.
- Learning rate. We experimented with various learning rates and found that 7.5×10^{-5} was a good learning rate. We experimented with higher learning rates and found that they made the training unstable.
- We use a hidden width of 1024 and did not experiment with other widths.
- We did not use batch normalization, keeping in line with the suggestions from Chapter 1.1.
- We always sampled the full 576 point grid corresponding to the default encodings, again keeping in line with the theory developed in Chapter 1.1.

During training we use a learning rate of 7.5×10^{-5} with the Adam optimizer using the default PyTorch configuration. Our MLP has four hidden layers and 48 Fourier features. We train for 100 epochs using a standard cosine learning rate scheduler. During training we use a quadtree with a 10% random merging strategy. The reasoning for this is two-fold. First, introducing the random merging allows the model to see more patch locations during training, and thus effectively increases the image sizes that our model can handle. Second, it acts as a regularizer to prevent overfitting to the data-distribution of our training dataset. This is because with a deterministic sampler the positional encodings would align themselves to common QtP patterns. For example, if the objects in the training data were centered and the background had low semantic content, the model may overfit to such a situation and not be robust to situations in which the objects of interest are not centered in the image.

4.3.4.3. *Final Training Curves.* The noise in the $L_{\texttt{[CLS]}}$ term and the grad norm terms is expected as a consequence of training on multiple resolutions simultaneously, as well as using the random selection strategy during training. We observed that the variance of these curves decreases if we restrict training to a narrower band of resolutions and / or remove the random selection from the training. The full training curve appears in Figure 4.5

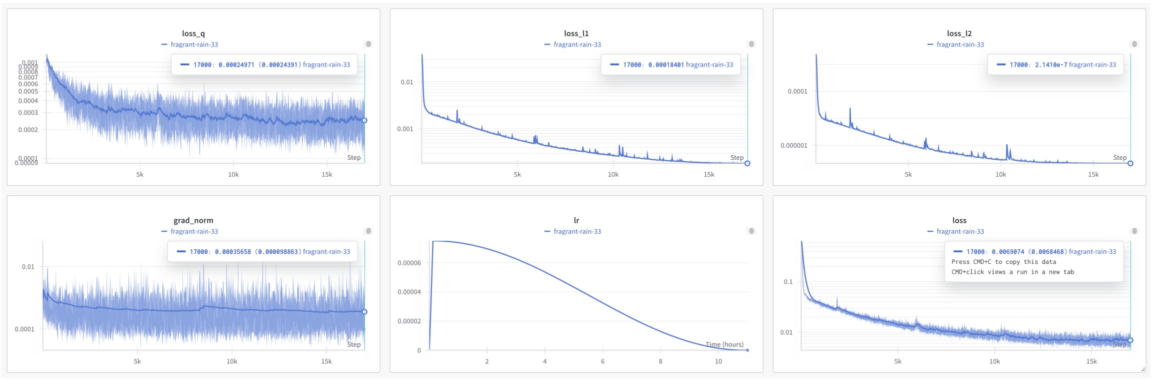


FIGURE 4.5. The training curves for our MLP training run. In the upper left is the $L_{\texttt{[CLS]}}$. The upper middle is the residual loss $\mathcal{R}$ from equation 4.3. The upper right is the L^2 analog of $\mathcal{R}$. Bottom left is the grad norm with respect the positional encodings, given by $\mathcal{B}_{\text{Interp}}$ in equation 4.1 above. The bottom middle is our learning rate as a function of time. The bottom right is the training loss, which is the sum of the upper left and upper middle panels. We did not stop training when spikes occurred and we found that the loss spikes were transient. The upper row is plotted with a logarithmic y-axis, as are the bottom left and bottom right panels.

4.4. Experimental Results

Recall that the parameter α from equation 4.1 controls the amount of pruning done to the quadtree. We perform sweeps in α and image size, over a suite of multi-modal benchmarks. For some benchmarks we additionally sweep native image resolution vs. cropped image resolutions. We report the best score from our sweeps in Table 4.1. We choose to look at the performance on V^* [108], MM-Bench [66], POPE [63], CV-Bench [97], the visual portion of ScienceQA [67], MME [43], and the RealWorld-QA benchmark [110]. We use VLM Eval [35] to do the evaluations on MM-Bench, POPE, ScienceQA, MME, and RealWorld-QA. We use a custom evaluation script to evaluate V^* and CV-Bench. More details of our experimental setup are contained in Chapter 4.4.1.1.

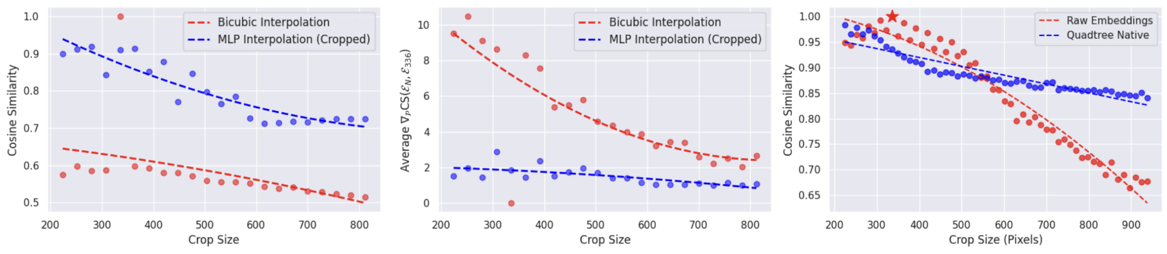


FIGURE 4.6. The first two panels compare of our MLP interpolation with bicubic interpolation. We plot $\mathcal{B}_{\text{Interp}}$ in the first panel as a measure of interpolation bias and $C^z_{N\to 336}$ in the middle panel as a measure of mesoscopic bias. The third panel shows a comparison between the [CLS] tokens of various image sizes with (blue) and without (red) QtP. All data is collected and averaged over the images from the V^* benchmark.

QLIP Reduces Measured Interpolation and Mesoscopic Bias: In Figure 4.6 we plot a comparison between QLIP and the vanilla CLIP encoder using bicubic interpolation, which we found outperformed bilinear interpolation. We see that MLP training successfully reduces interpolation bias as measured by $\mathcal{B}_{\text{Interp}}$, and brings the cosine similarity between the `[CLS]` tokens together as predicted by our theoretical assumptions. Next, we observe that the quadtree selection mechanism mitigates the effects of mesoscopic bias by slowing the rate at which the cosine similarity of the CLIP `[CLS]` tokens diverge as a function of image size (Figure 4.6, rightmost panel).

TABLE 4.1. Performance comparison between LLaVA-QLIP and baseline LLaVA models. **Bold** highlights the better-performing variant of the same base model. Underlining denotes the best result across all models. An asterisk (*) indicates results obtained using cropped images. Performance increases and decreases are annotated in green and red, respectively.

Model	V^*	MM-Bench	POPE F1	CV-Bench	Sci-QA	MME	RW-QA
LLaVA-1.5-7b	42.4	**62.5**	74.4	39.9	**64.0**	1207	**49.0**
+ QLIP	**53.4**	59.7	**79.6**	**40.2**	63.5	**1241**	47.3
	(+11.0)	(-2.8)	(+5.2)	(+0.3)	(-0.5)	(+34)	(-1.7)
LLaVA-1.5-13B	45.0	67.4	82.4	<u>**61.6**</u>	67.8	<u>**1390**</u>	48.0
+ QLIP	<u>**58.6**</u>	<u>**67.9**</u>*	<u>**83.6**</u>	60.7*	<u>**67.9**</u>	1388*	<u>**49.4**</u>
	(+13.6)	(+0.5)	(+1.2)	(-0.9)	(+0.1)	(-2)	(+1.4)

QLIP Significantly Improves the Detailed Visual Grounding on High-Resolution Images: The V^* benchmark [108] is a challenging, vision centric benchmark focused on fine-grained image understanding. This benchmark is particularly challenging for CLIP-based vision encoders because the questions are designed to be answered with access to the full-resolution image (see Figure 4.1). Without access to all of the appropriate visual information the model is often reduced to guessing.

Figure 4.7 demonstrates that in the absence of the quadtree selection method, our MLP interpolation network already allows the model to effectively utilize all of the image tokens from the original image. We note that the 7B parameter model seems robust to MLP interpolation on image sizes which were not seen during training, while the 13B parameter model is much more sensitive to interpolation error beyond the training regime. These results already indicate that there is a large performance gap that can be closed with minimal interventions, indicating that a significant portion of the poor performance on high-resolution image tasks can be explained simply by a lack of access to high-quality visual input signal (c.f. [61]). This result indicates that the CLIP encoder and LLaVA weights possess sufficient capacity to do VQA, but lack high-quality inputs.

Figure 4.8 shows the full sweep over image size and α, plotted with cubic best-fit lines. The x-axis is measured in percentage of tokens seen compared to the baseline model, on a logarithmic scale. Because our method is content-aware, this is a better x-axis scale than directly plotting α. We see a clear trend where increasing α (i.e. pruning the quadtree and decreasing the amount of image tokens seen) increases performance with maximal performance occurring for $\alpha > 0$. This indicates that the QtP mechanism is complementing the MLP interpolation to boost VQA performance, either by reducing the number of image tokens and sending stronger attention signal to the LLM [60,98], by reducing noise by combining redundant image patches through merging, or both. Our ablations in Section 4.5 below suggest that the latter is more likely.

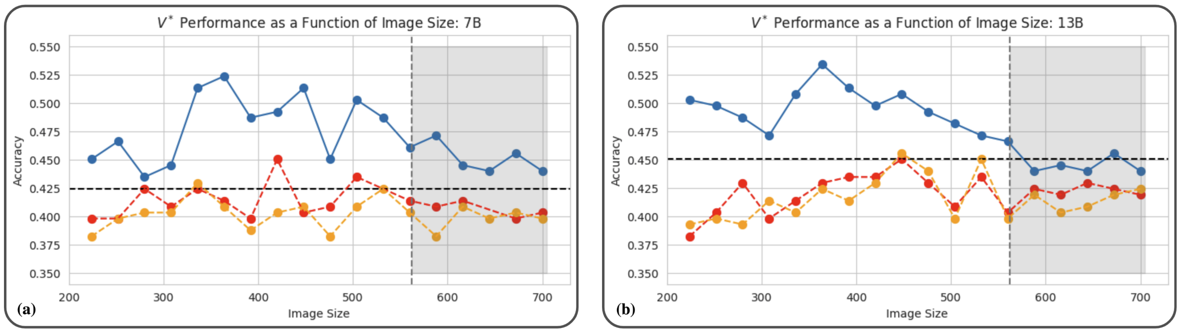


FIGURE 4.7. The performance on V^* using re-scaled and cropped images with no quadtree selection mechanism and our MLP interpolation. The red line is with bicubic interpolation and the orange line is with bilinear interpolation. The black line represents performance of the base CLIP model with 336×336 cropping. The 7B model is plotted on the left, and the 13B model on the right. We see that neither bilinear nor bicubic interpolation is suitable for extending CLIP to larger resolutions.

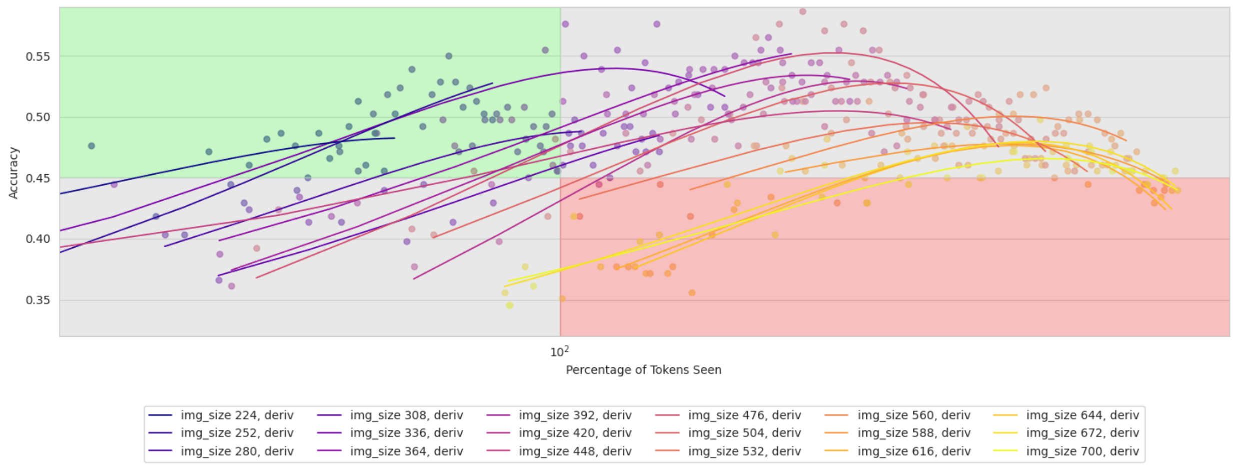


FIGURE 4.8. The compute vs. accuracy curves for our sweep of V^* with the LLaVA-QLIP-13B model. The x-axis is on a logarithmic scale. The green-shaded region highlights experiments where our model **surpasses the baseline** with **fewer** visual tokens.

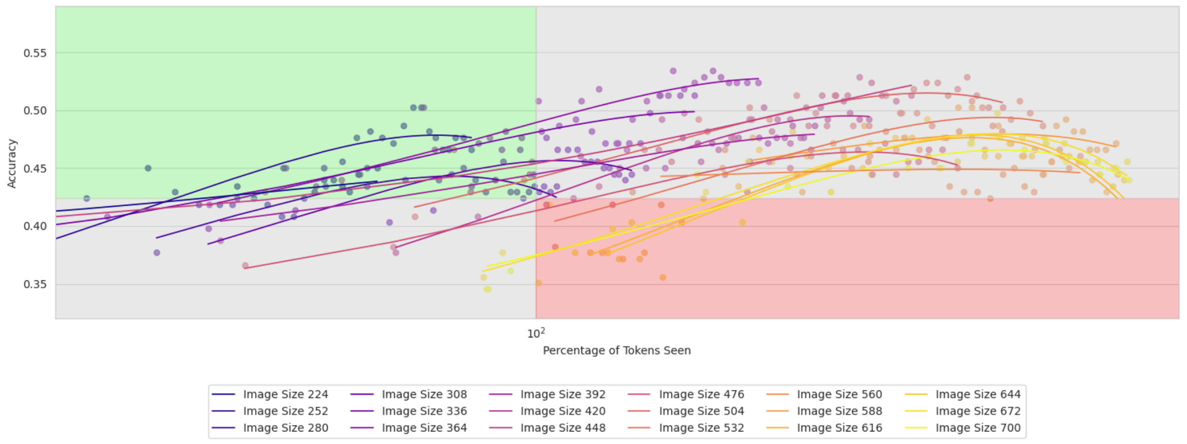


FIGURE 4.9. The compute vs. accuracy curves for. our sweep of V^* with the LLaVA-QLIP-7B model. The x-axis is on a logarithmic scale. The green-shaded region highlights experiments where our model **surpasses the baseline** with **fewer** visual tokens.

Improved Token Efficiency: Previous studies that reduce token counts have aimed at matching MLLM performance with fewer tokens [18, 19, 52, 61, 95]. Our work is largely orthogonal to the aforementioned works, however we note in Figures 4.8 and 4.9 that we can achieve higher than baseline accuracy with *fewer* image tokens than the baseline model. This is shown in the figure by the region shaded in green, which represents higher than baseline accuracy with lower than baseline numbers of tokens. This reveals that the quadtree selection method, which is responsible for pruning tokens, is doing so in a way that provides higher-quality visual signal to the LLM. The work [61] demonstrated that such improved performance with reduced tokens is theoretically practical, but to our knowledge this work is the first time such a result has been achieved in practice.

QLIP Matches or Improves Performance Across a Range of MLLM Benchmarks: Because our method is trained to be both minimally invasive and not require re-training of the MLLM, we can adjust the model parameters to fit the task at hand without re-training. Because our training program was oriented towards matching CLIP outputs on images which are the same size as CLIP was trained on, we can nearly achieve baseline performance for any benchmark by using 336×336 images with $\alpha = 0$. Any loss in performance beyond that can be attributed to the error in interpolating the CLIP embeddings with our MLP network. Notably we find little to no change in performance on MM-Bench, CV-Bench, Sci-QA, MME, or RealWorld QA. Hu et al. [52] were able to match LLaVA performance on Sci-QA with only two image tokens, which we suspect indicates that the performance on Sci-QA is almost entirely dependent on the `[CLS]` token, not on any fine-grained image encoding.

LLaVA 13B is Sensitive to Image Aspect Ratio: On three of the seven benchmarks the 13B model attained its best performance when the input images were cropped to be square at the original image resolution of 336×336 with $\alpha < 0.1$. We found that performance quickly dropped off for these three benchmarks when we varied image size or increased α. We suspect that the 13B parameter version of LLaVA is much more sensitive to deviations in the `[CLS]` token, and the drop-off in performance seems correlated with the change in cosine similarity of the `[CLS]` token plotted in Figure 4.6. We did not observe the same trend in the 7B model, nor did we observe this trend on V^*, where the content of the `[CLS]` token is not helpful for answering the questions.

TABLE 4.2. Comparison of LLaVA-QLIP with other models which improve fine-detail grounding. We report the numbers from the authors' papers. Note that S^2 requires pre-training and instruction tuning of the LLM [89], and that SEAL requires fully re-placing the vision encoder before pre-training and instruction tuning [108].

Model	V^*-Att	V^*-Rel	V^* Overall	POPE F1
QLIP-7B	50.4	60.5	53.4	79.6
S^2-7B [89]	51.3	61.8	55.5	-
QLIP-13B	53.9	65.8	58.6	**83.6**
S^2-13B [89]	50.4	63.2	55.5	-
SEAL (7B) [108]	**74.8**	**76.3**	**75.4**	82.4

Hallucination can be Mitigated by Reducing the Number of Image Tokens: The POPE dataset was designed to measure the hallucination proclivity of MLLMs [63]. The proposed measurement of model performance for POPE is the F1 score. For both the 7B and 13B QLIP models we saw increased performance on POPE, with more significant gains for the 7B model. In fact, QLIP even outperforms SEAL [108] which is a heavily optimized version of LLaVA designed specifically to address fine-grained VQA (see Table 4.2). We found that peak POPE performance occurred with the smallest image size we tested (shortest edge is 224 pixels), and an $\alpha = 0.7$, corresponding to slightly less than 50% of the baseline image tokens. We plot the performance of QLIP on the POPE dataset in Figure 4.10. We see from this figure that for the 7B model, smaller images reduce hallucination universally, while for the 13B model the performance begins to decrease after a certain point. Interestingly, 252 seems to be the best image resolution for both models, however the 13B model under-performs the baseline for image sizes of 224, while the 7B model has its strongest performance at this image size. We note that this is further evidence of the sensitivity of the 13B model to its training image size.

One possible reason for the reduced hallucination may simply be a reduced number of image tokens being fed to the model. Recent works [60,98] suggest that having too many tokens can lead to degradations in performance.

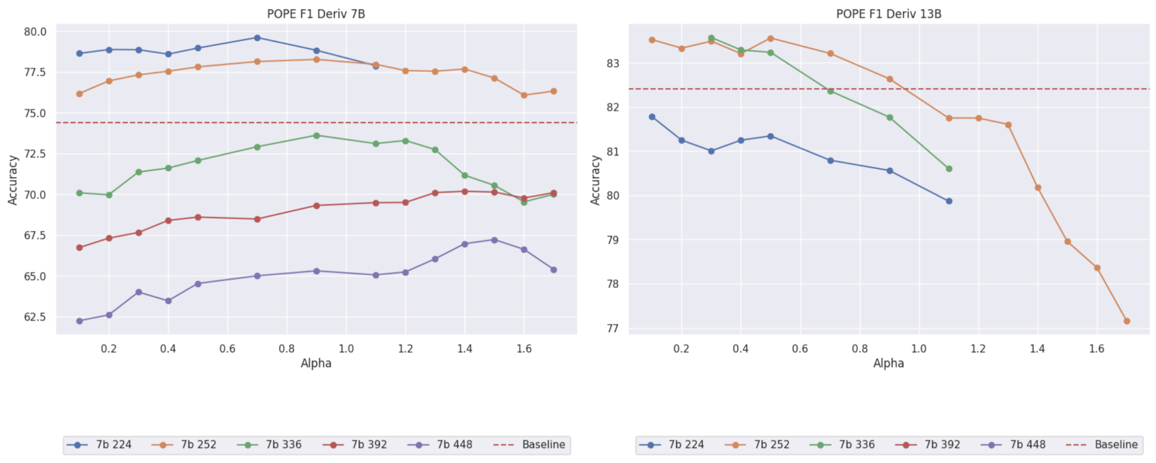


FIGURE 4.10. Sweeps on the POPE benchmark [63]. Baselines are indicated by the dashed red line. Smaller resolution images seem to result in fewer hallucinations.

4.4.1. Extended Experimental Results. This section catalogs some additional experiments whose results can be found populating Table 4.1. We note that our method is engineered specifically for improving performance on vision-centric, high-resolution, benchmarks like V^*. Recent work has also shown that MLLMs continue to perform well on some benchmarks which claim to measure visual capabilities even when the visual information is removed [97].

4.4.1.1. *More Details on Evaluation Strategies.* We chose parameter sweep ranges through ad-hoc probing of both image size and α values. Once we found good endpoints we would sweep the values in-between, keeping the cost of evaluations in mind as we chose our sweep parameters. We would stop sweeping early if the results were trending in the wrong direction, since over regularization from QtP is expected to cause consistent declines in performance after a certain threshold. Our primary goal with these sweeps was to understand the ways in which the performance of our algorithm depends on the hyperparameters.

4.4.1.2. *Native Image Resolution vs. Cropped.* For models that did not perform well-enough using the native resolution we would switch to sweeping the cropped versions. In one case we report numbers from our model with no QtP and only the MLP interpolation (MME). We found that giving the model images at the native aspect ratio sometimes led to degraded performance in some cases. We hypothesize that there are two reasons for this. First, while our MLP network does reduce the effects of interpolation bias, there continues to be some error from our interventions. In cases where the model is highly sensitive to the input image positions, this small error can continue to cause errors to propagate. Second, we have reason to suspect that there continue to be other biases in the MLLMs vision pipeline, as we have observed in several places in this chapter that the models, especially the 13B model, seem particularly sensitive to image resolution. We leave further investigations of these hypotheses to future work.

Interestingly, our results from the POPE benchmark [**63**] (Figure 4.10) indicate that the cause of the decreased performance on the native image resolutions is unlikely to be caused by hallucination.

4.4.1.3. *MME Benchmark.* We found that performance with native images was poor on MME so we swept cropped images. For the 7B model this lead to overperformance of the baseline, but for the 13B model we could not get overperformance of the baseline with $\alpha > 0$. Our top performing 13B model was with 336 image size, random selection, and $\alpha = 0$. This represents our model's closest approximation to the baseline model and any error is accounted for by the numerical error in the MLP interpolation procedure. MME evaluations are expensive and we were not able to perform more comprehensive sweeps.

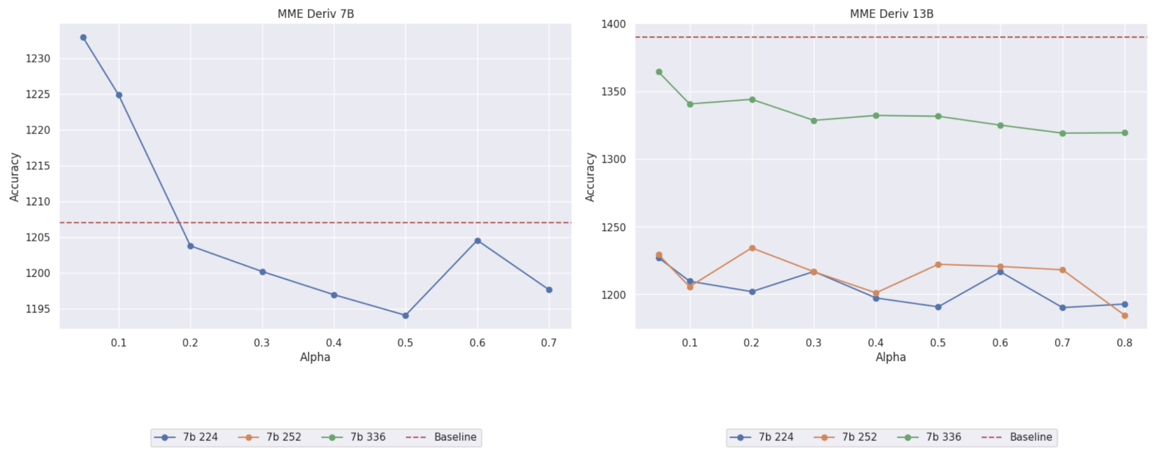


FIGURE 4.11. Sweeps on the MME benchmark [**43**]. Baselines are indicated by the dashed red line.

4.4.1.4. *MM-Bench.* For MM-Bench we found that the results were better with cropping than using the native image resolution. We swept several image sizes, but pruned the sweeps if the performance was proving poor. We swept image sizes of $224, 252, 336, 392$ and 448. The results of our sweeps, including the specific α values chosen for each image size are shown in Figure 4.12.

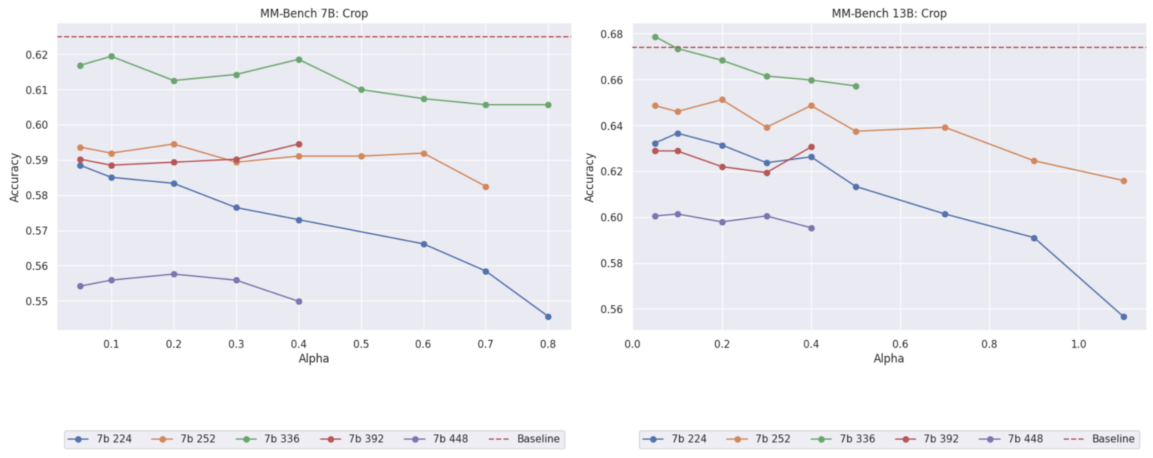


FIGURE 4.12. Sweeps on the MMBench benchmark [66]. Baselines are indicated by the dashed red line.

4.4.1.5. *SciQA.* The ScienceQA benchmark consists of science related question answering based on science related images [67]. Our results for the ScienceQA benchmark can be found in Figure 4.13. Previous authors have been able to remove large numbers of image tokens and continue to maintain near-baseline performance on the ScienceQA benchmark [52], and we find that we achieve similar results. Notably, unlike the Matryoshka model proposed in [52], our method can actually **outperform** the baseline 13B model using fewer tokens than the baseline model. However we find that in general performance is roughly maintained at a slightly worse than baseline level as we prune tokens.

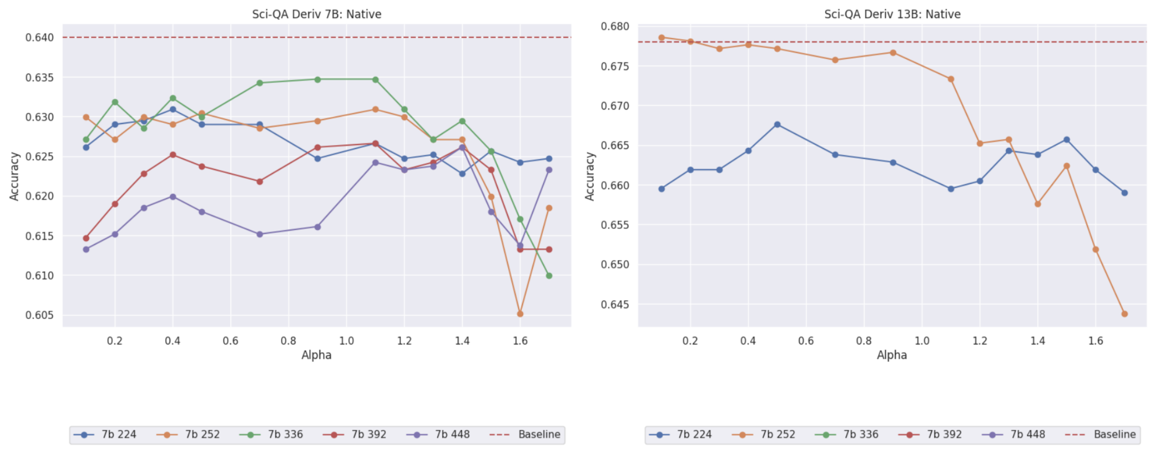


FIGURE 4.13. Sweeps on the ScienceQA benchmark [67]. Baselines are indicated by the dashed red line.

4.4.1.6. *RealWorldQA.* We are able to perform fairly comprehensive sweeps on the RealWorldQA benchmark [110] as the benchmark proves inexpensive to evaluate. The results of our sweeps on the 7B and 13B

QLIP model are shown in Figure 4.14. Similar to ScienceQA (Section 4.4.1.5), the 13B model outperforms the baseline while the 7B model cannot. We find that performance is largely insensitive to the number of image tokens, but does depend on the image size, suggesting a possible over-reliance on some mesoscopic features of the dataset. Our results on RealWorldQA can be found in Figure 4.14.

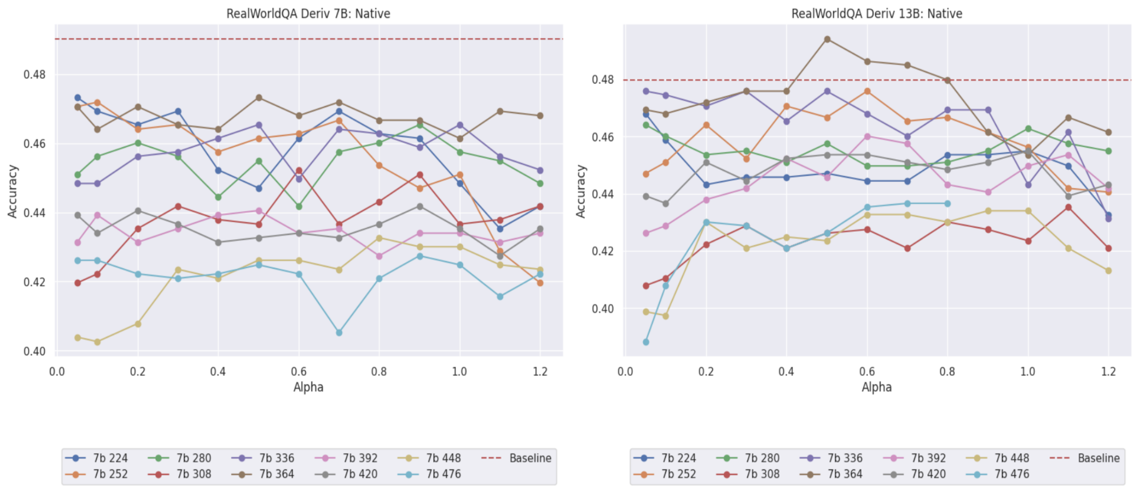


FIGURE 4.14. Sweeps on the RealWorldQA benchmark [110]. Baselines are indicated by the dashed red line.

4.4.1.7. *CV-Bench.* CV-Bench [97] is expensive to run sweeps over. Because of this we searched a relatively small percentage of the search space. We found that performance using the 7B model was best for the larger image sizes (see Figure 4.15). We found that the QtP procedure typically led to decreasing performance on CV-Bench, and a preliminary sweep showed us that the 7B model performed best when using the larger image sizes.

For 13B the model performed poorly with the native image sizes and we swept crops instead. For this sweep we found smaller images were better, with peak performance occurring when the images matched the pre-training image size, 336×336 (see Figure 4.16).

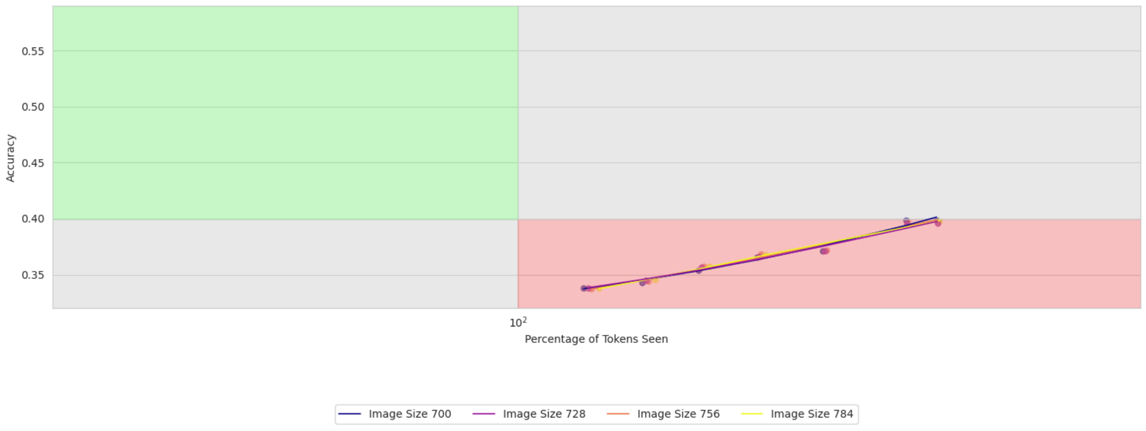


FIGURE 4.15. Compute vs. accuracy curves for our sweeps of CV-Bench, 7B, native resolution.

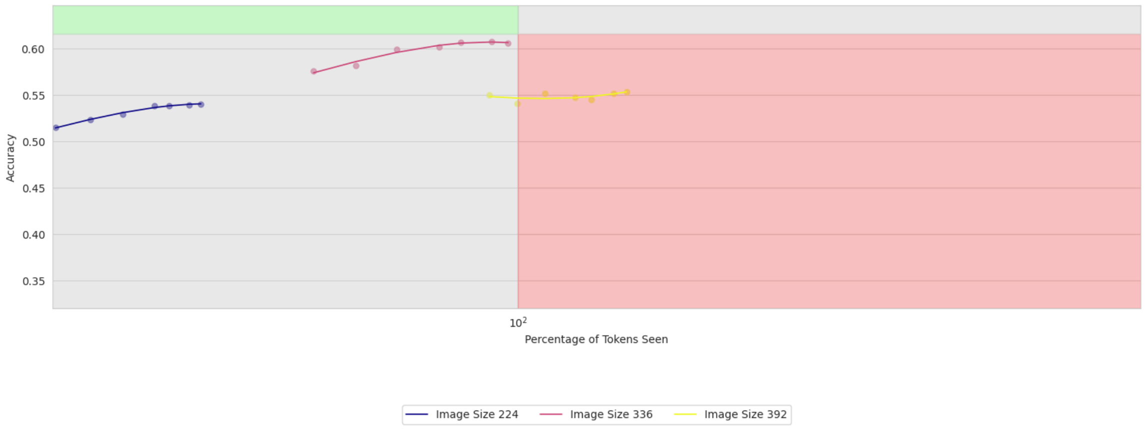


FIGURE 4.16. Compute vs. accuracy curves for our sweeps of CV-Bench, 13B, cropped resolution.

4.4.1.8. *V^*-Bench.* For the V^* benchmark [108] we sweep image size and α using our native image resolutions. We did not sweep V^* using cropped images. We sweep image sizes between 224 and 700 in steps of 28. For the derivative selection strategy we sweep $\alpha \in (0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.7, 1.9, 2.$ For the random selection strategy we sweep $\alpha \in (0.0, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6)$. V^* evaluations are the least expensive of our chosen evaluations and therefore we have the most comprehensive sweeps on this benchmark.

We include the results of our sweeps in color coded tables below. The sweep for the 7B model with the derivative selection strategy can be found in Table 4.3. The sweep for the 7B model with the random selection strategy can be found in Table 4.4. The sweep for the 13B model with the derivative selection

TABLE 4.3. Accuracy on V^* for LLaVA-QLIP-7B using derivative selection method. Native resolution.

Image Size	$\alpha = 0.05$	$\alpha = 0.10$	$\alpha = 0.20$	$\alpha = 0.30$	$\alpha = 0.40$	$\alpha = 0.50$	$\alpha = 0.60$	$\alpha = 0.70$	$\alpha = 0.80$	$\alpha = 0.90$	$\alpha = 1.00$	$\alpha = 1.10$	$\alpha = 1.20$	$\alpha = 1.30$	$\alpha = 1.40$	$\alpha = 1.50$	$\alpha = 1.70$	$\alpha = 1.90$	$\alpha = 2.10$	$\alpha = 2.50$	$\alpha = 3.00$
224	**45.03%**	**45.03%**	44.50%	42.93%	43.46%	43.98%	43.98%	42.93%	42.41%	40.84%	40.84%	41.88%	42.41%	43.46%	41.88%	41.88%	**45.03%**	42.41%	43.46%	37.17%	39.27%
252	46.60%	47.64%	47.64%	46.60%	46.60%	47.64%	48.17%	**50.26%**	**50.26%**	48.69%	47.12%	47.64%	48.17%	47.64%	45.55%	45.03%	45.03%	41.88%	42.93%	40.31%	34.55%
280	43.46%	42.93%	43.46%	42.93%	43.46%	42.93%	42.41%	42.93%	42.93%	**45.55%**	43.98%	42.93%	43.98%	45.03%	44.50%	43.98%	43.98%	43.46%	43.46%	42.41%	37.70%
308	44.50%	43.98%	44.50%	44.50%	45.03%	45.03%	45.55%	45.55%	45.55%	46.60%	46.60%	46.60%	47.64%	44.50%	46.60%	**48.17%**	43.46%	41.36%	40.31%	41.36%	39.79%
336	**51.31%**	**51.31%**	**51.31%**	**51.31%**	49.74%	47.12%	47.12%	47.12%	47.12%	48.17%	49.21%	48.69%	48.17%	47.64%	46.60%	47.12%	50.26%	44.50%	45.03%	40.84%	36.65%
364	52.36%	52.36%	52.36%	52.36%	52.88%	**53.40%**	52.88%	52.36%	51.83%	**53.40%**	51.31%	50.79%	49.21%	47.12%	51.83%	50.79%	50.79%	47.64%	46.60%	43.46%	41.88%
392	**48.69%**	47.64%	**48.69%**	47.64%	47.64%	47.12%	47.64%	45.55%	47.64%	47.12%	47.12%	47.12%	47.64%	45.03%	45.03%	46.60%	46.07%	44.50%	42.93%	43.98%	38.74%
420	49.21%	49.21%	49.74%	50.79%	**51.31%**	49.21%	50.26%	46.60%	47.64%	47.12%	49.21%	50.26%	49.21%	47.64%	46.60%	45.55%	49.21%	45.03%	42.41%	40.84%	37.70%
448	51.31%	50.79%	51.31%	52.36%	**52.88%**	51.31%	51.31%	51.31%	50.26%	49.21%	48.17%	49.21%	47.64%	48.17%	49.74%	46.60%	43.46%	45.55%	46.60%	42.93%	39.79%
476	45.03%	46.60%	46.60%	45.55%	45.55%	44.50%	47.12%	44.50%	46.60%	46.60%	**48.69%**	**48.69%**	45.03%	45.55%	42.93%	44.50%	45.55%	41.88%	41.88%	38.22%	36.65%
504	50.26%	50.26%	51.31%	**52.88%**	51.83%	51.31%	52.36%	52.36%	51.83%	49.74%	51.31%	48.69%	48.69%	51.31%	49.74%	51.31%	50.26%	49.21%	46.60%	43.98%	40.84%
532	48.69%	49.74%	**50.79%**	49.21%	48.17%	48.69%	50.26%	**50.79%**	48.17%	47.64%	48.69%	49.74%	49.74%	48.17%	45.55%	45.03%	46.07%	49.21%	46.60%	42.93%	38.22%
560	46.07%	45.03%	46.60%	45.55%	44.50%	42.93%	43.46%	45.03%	44.50%	42.41%	42.93%	42.93%	43.46%	45.55%	46.60%	**47.64%**	46.07%	47.12%	44.50%	45.03%	41.88%
588	47.12%	46.60%	46.60%	46.60%	48.17%	48.17%	49.21%	48.69%	47.12%	46.60%	46.07%	47.12%	48.69%	46.60%	**50.26%**	**50.26%**	46.07%	46.60%	48.69%	42.93%	47.64%
616	44.50%	42.93%	43.98%	45.55%	46.07%	44.50%	46.07%	46.07%	**47.64%**	**47.64%**	**47.64%**	46.60%	46.07%	46.07%	45.55%	43.46%	40.31%	37.70%	37.17%	37.70%	37.70%
644	43.98%	43.46%	44.50%	45.55%	46.07%	45.55%	46.07%	47.12%	46.07%	45.55%	**49.21%**	47.64%	45.03%	45.03%	48.17%	46.60%	42.41%	35.60%	37.17%	37.17%	37.70%
672	45.55%	45.03%	45.55%	45.55%	47.12%	47.64%	47.64%	**49.74%**	47.12%	45.03%	47.64%	47.12%	42.93%	46.07%	45.55%	42.93%	40.31%	37.70%	35.08%	37.70%	35.60%
700	43.98%	45.03%	44.50%	45.55%	45.55%	47.12%	46.60%	**47.64%**	47.12%	45.03%	45.03%	44.50%	42.93%	43.46%	40.31%	44.50%	39.79%	41.88%	36.13%	34.55%	34.55%

TABLE 4.4. Accuracy on V^* for LLaVA-QLIP-7B using random selection method. Native resolution.

Image Size	$\alpha = 0.00$	$\alpha = 0.05$	$\alpha = 0.10$	$\alpha = 0.20$	$\alpha = 0.30$	$\alpha = 0.40$	$\alpha = 0.50$	$\alpha = 0.60$
224	**46.07%**	43.46%	40.31%	40.84%	40.84%	40.31%	34.03%	35.60%
252	45.03%	**47.12%**	45.55%	41.36%	40.84%	35.60%	36.65%	35.08%
280	43.98%	44.50%	43.46%	**45.03%**	39.79%	40.31%	42.93%	40.84%
308	**45.55%**	42.41%	41.36%	40.84%	37.70%	37.70%	38.74%	37.17%
336	**51.31%**	46.60%	45.55%	43.98%	41.36%	35.60%	38.74%	37.17%
364	**52.88%**	47.64%	51.83%	44.50%	45.03%	36.13%	44.50%	38.74%
392	48.69%	**49.21%**	46.60%	42.93%	42.41%	41.88%	43.46%	33.51%
420	**49.74%**	47.12%	47.64%	47.12%	48.17%	37.17%	36.65%	37.70%
448	**52.88%**	50.79%	45.55%	43.46%	40.84%	37.17%	39.27%	38.22%
476	**46.07%**	45.03%	42.93%	43.98%	42.93%	40.84%	33.51%	35.08%
504	**49.74%**	47.64%	48.17%	**49.74%**	42.41%	44.50%	37.70%	40.31%
532	48.17%	**49.21%**	47.12%	48.17%	42.93%	42.41%	39.27%	41.88%
560	46.07%	45.55%	43.98%	**48.69%**	41.88%	42.41%	43.46%	42.93%
588	47.64%	46.60%	**49.74%**	45.55%	49.21%	46.07%	43.98%	46.07%
616	44.50%	44.50%	41.88%	**45.55%**	39.79%	38.74%	40.31%	37.70%
644	42.93%	46.07%	45.55%	**48.17%**	35.60%	42.41%	39.79%	37.70%
672	45.03%	45.03%	**47.64%**	41.88%	39.27%	39.27%	39.27%	34.03%
700	43.46%	**46.60%**	46.07%	39.79%	41.88%	40.31%	39.79%	35.08%

strategy can be found in Table 4.5. The sweep for the 13B model with the random selection strategy can be found in Table 4.6.

TABLE 4.5. Accuracy on V^* for LLaVA-QLIP-13B using derivative selection method. Native resolution.

Image Size	$\alpha = 0.05$	$\alpha = 0.10$	$\alpha = 0.20$	$\alpha = 0.30$	$\alpha = 0.40$	$\alpha = 0.50$	$\alpha = 0.60$	$\alpha = 0.70$	$\alpha = 0.80$	$\alpha = 0.90$	$\alpha = 1.00$	$\alpha = 1.10$	$\alpha = 1.20$	$\alpha = 1.30$	$\alpha = 1.40$	$\alpha = 1.50$	$\alpha = 1.70$	$\alpha = 1.90$	$\alpha = 2.10$	$\alpha = 2.50$	$\alpha = 3.00$
224	50.26%	45.55%	48.69%	46.07%	**51.31%**	49.21%	47.64%	46.60%	48.69%	45.03%	47.64%	48.69%	48.17%	46.07%	44.50%	47.12%	41.88%	47.64%	42.93%	41.36%	35.60%
252	49.74%	50.26%	51.31%	52.36%	50.79%	52.88%	**54.97%**	52.88%	51.83%	49.21%	53.93%	52.36%	51.83%	50.26%	45.55%	47.12%	43.98%	42.93%	40.31%	39.27%	36.65%
280	48.69%	49.74%	48.69%	46.07%	45.55%	50.26%	49.21%	47.64%	49.21%	48.17%	47.12%	**50.79%**	50.26%	49.74%	47.64%	47.12%	44.50%	43.98%	41.36%	42.41%	40.31%
308	47.12%	48.69%	47.12%	48.17%	47.12%	**49.21%**	**49.21%**	47.64%	45.03%	48.69%	46.07%	47.64%	47.64%	45.55%	47.12%	45.03%	42.93%	41.36%	39.79%	41.36%	36.65%
336	50.79%	50.79%	51.83%	53.40%	52.88%	52.36%	**57.59%**	54.45%	52.36%	55.50%	53.93%	54.97%	**57.59%**	55.50%	50.79%	49.74%	49.74%	48.69%	43.46%	44.50%	37.70%
364	53.40%	55.50%	**56.54%**	55.50%	54.45%	54.45%	54.97%	53.93%	53.93%	53.93%	53.93%	50.26%	48.17%	50.26%	50.26%	50.26%	51.31%	49.74%	47.64%	41.88%	38.74%
392	51.31%	**55.50%**	53.40%	54.45%	54.45%	52.88%	51.31%	54.45%	52.36%	54.45%	50.79%	50.26%	52.36%	51.83%	51.31%	51.83%	51.83%	46.07%	44.50%	42.93%	36.13%
420	49.74%	52.88%	53.40%	52.88%	**55.50%**	52.88%	53.40%	52.88%	51.31%	52.88%	51.83%	51.31%	52.88%	52.88%	51.83%	49.74%	48.17%	46.07%	44.50%	39.79%	37.70%
448	50.79%	49.74%	50.79%	**51.31%**	47.64%	47.12%	48.69%	**51.31%**	**51.31%**	49.21%	**51.31%**	**51.31%**	50.26%	47.64%	48.17%	50.79%	48.69%	48.17%	45.55%	42.41%	37.70%
476	49.21%	51.31%	51.31%	49.74%	49.21%	49.74%	51.83%	52.36%	52.36%	55.50%	57.07%	57.59%	57.07%	**<u>58.64%</u>**	57.59%	56.54%	49.74%	45.55%	47.12%	40.84%	39.27%
504	48.17%	48.17%	49.74%	51.31%	49.74%	50.79%	51.83%	49.74%	50.79%	49.74%	51.83%	53.40%	52.88%	52.88%	**54.45%**	52.36%	52.36%	50.26%	48.69%	41.88%	40.31%
532	47.12%	45.55%	48.69%	48.69%	46.07%	46.60%	46.60%	48.69%	48.17%	49.21%	48.69%	48.69%	48.69%	49.21%	49.21%	50.26%	49.74%	**51.31%**	47.64%	44.50%	41.88%
560	46.60%	47.64%	48.69%	49.74%	50.26%	50.26%	49.74%	49.21%	**52.36%**	51.83%	51.83%	48.69%	48.17%	49.21%	46.07%	47.64%	48.17%	51.31%	48.69%	47.64%	41.88%
588	43.98%	43.98%	44.50%	47.12%	46.60%	48.69%	48.17%	46.07%	46.07%	**49.74%**	45.55%	45.03%	46.60%	46.07%	46.60%	47.64%	43.98%	47.12%	42.93%	41.36%	41.88%
616	40.84%	42.41%	43.46%	40.84%	46.07%	45.03%	45.55%	47.12%	46.60%	**49.74%**	47.64%	46.60%	47.12%	47.64%	42.41%	43.98%	37.70%	40.31%	37.17%	37.70%	38.22%
644	42.41%	42.41%	43.98%	43.98%	42.93%	45.55%	**46.60%**	44.50%	46.07%	**46.60%**	45.03%	45.55%	43.98%	45.55%	44.50%	43.46%	44.50%	42.41%	39.27%	36.65%	37.17%
672	41.88%	45.03%	43.46%	42.93%	43.98%	47.12%	47.12%	46.60%	**49.21%**	48.17%	47.64%	46.60%	43.98%	47.12%	47.64%	45.03%	43.98%	40.84%	38.74%	38.22%	35.08%
700	43.46%	42.41%	46.07%	45.03%	43.98%	**47.12%**	43.98%	46.07%	45.55%	46.07%	**47.12%**	44.50%	46.07%	44.50%	42.41%	44.50%	45.03%	40.31%	38.22%	36.65%	35.60%

TABLE 4.6. Accuracy on V^* for LLaVA-QLIP-13B using random selection method. Native resolution.

Image Size	$\alpha = 0.00$	$\alpha = 0.05$	$\alpha = 0.10$	$\alpha = 0.20$	$\alpha = 0.30$	$\alpha = 0.40$	$\alpha = 0.50$	$\alpha = 0.60$
224	49.21%	45.55%	**51.83%**	43.46%	45.55%	39.27%	38.22%	36.65%
252	**49.74%**	42.41%	46.60%	44.50%	48.17%	38.22%	36.13%	39.27%
280	**48.17%**	45.55%	42.41%	42.93%	40.84%	38.74%	37.70%	39.27%
308	**49.21%**	44.50%	45.55%	43.46%	42.41%	41.36%	37.17%	34.03%
336	51.31%	**<u>55.50%</u>**	48.17%	45.55%	41.36%	40.31%	39.79%	40.31%
364	**53.93%**	51.83%	50.79%	45.55%	41.88%	38.74%	39.79%	38.74%
392	**53.40%**	**53.40%**	48.69%	45.03%	44.50%	36.65%	39.79%	40.31%
420	50.26%	48.69%	**52.88%**	47.12%	48.69%	42.41%	40.84%	34.55%
448	50.26%	**51.83%**	45.03%	43.98%	46.07%	41.36%	40.84%	37.70%
476	**49.74%**	47.64%	46.60%	48.17%	47.12%	47.12%	39.27%	40.84%
504	49.21%	48.17%	**49.74%**	45.55%	45.03%	36.65%	42.41%	36.65%
532	48.17%	48.69%	48.17%	**50.26%**	44.50%	42.93%	40.31%	40.31%
560	**46.60%**	45.55%	46.07%	46.07%	46.07%	42.41%	43.46%	44.50%
588	45.03%	43.46%	45.55%	**48.69%**	45.55%	43.98%	44.50%	45.03%
616	38.22%	42.93%	**45.55%**	44.50%	43.98%	41.36%	40.84%	38.22%
644	40.84%	40.84%	**47.64%**	41.88%	40.31%	40.31%	39.79%	38.22%
672	39.79%	42.41%	**42.93%**	41.36%	39.27%	**42.93%**	42.41%	37.17%
700	41.36%	40.31%	42.41%	42.93%	**43.98%**	39.27%	40.31%	35.08%

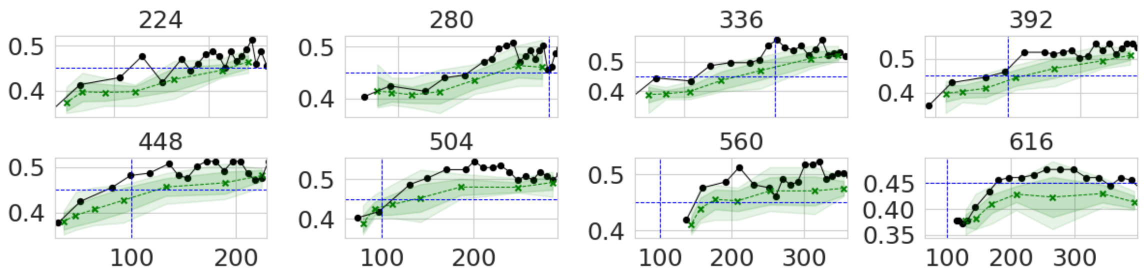


FIGURE 4.17. Ablation on V^* with QLIP-13B. The black curves are QLIP, with derivative pruning, and the green curves are QLIP with random pruning. The green curves are plotted with min/max lightly shaded, and the first standard deviation more darkly shaded. Each of the evaluations with the random selection strategy was run 10 times to compute the average and standard deviation. The x-axis is the percentage of image tokens seen compared to baseline, and the y-axis is accuracy. Each pane is labeled with its image size and the vertical and horizontal blue dashed lines represent baseline number of image tokens and baseline accuracy respectively.

We ablate our design decisions along two axes. The first axis is along interpolation strategy, where we show that our MLP network vastly outperforms bilinear and bicubic interpolation. Next, we demonstrate that our performance improvements from the quadtree mechanism are predicated on selection strategy and not due solely to a reduced token counts.

MLP Interpolation is Essential for Generalizing to Arbitrary Image Sizes: We experiment with using bicubic interpolation to scale evaluation with image size. We find that across all of our benchmarks bicubic and bilinear interpolation under-perform our MLP interpolation. This is clearly demonstrated for V^* bench in Figure 4.7, where the bicubic and bilinear interpolation schemes under-perform even the baseline model performance on average.

Performance Gains are not Solely a Result of a Reduced Number of Image Tokens: We verify that the derivative selection strategy provides a meaningful information signal to the downstream LLM by comparing it to using a random selection strategy which prunes quadtree branches at some random rate. We compare the performance of these two selection strategies on the V^* benchmark in Figure 4.17, where keeping precise semantic information about particular regions of the image is critically important. We find that on average there are large gaps in performance between random selection and derivative selection, indicating that our derivative selection strategy provides a more meaningful visual signal to the model.

4.5.1. Detailed Ablations.

We plot a more comprehensive ablation sweep over V^* than was provided in Figure 4.17 above. Figure 4.18 is the ablation for the 7B model on all of the image sizes and values of α

that we tested. Figure 4.19 is the ablation for the 13B model on all of the image sizes and values of α that we tested.

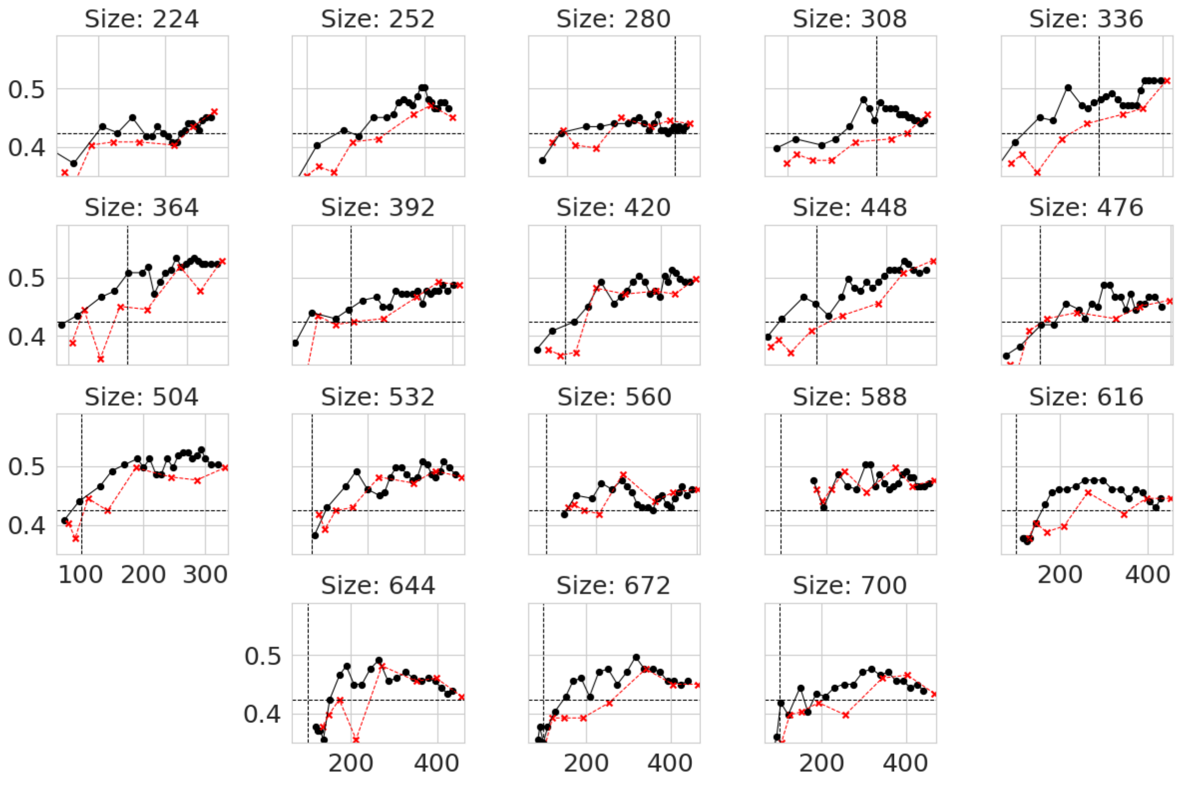


FIGURE 4.18. Ablation across a diverse range of image sizes of the QLIP-7B model on the V^* dataset. The Black line is the QLIP performance with the derivative selection strategy and the red line is a random selection strategy. Each random selection trial was only run once.

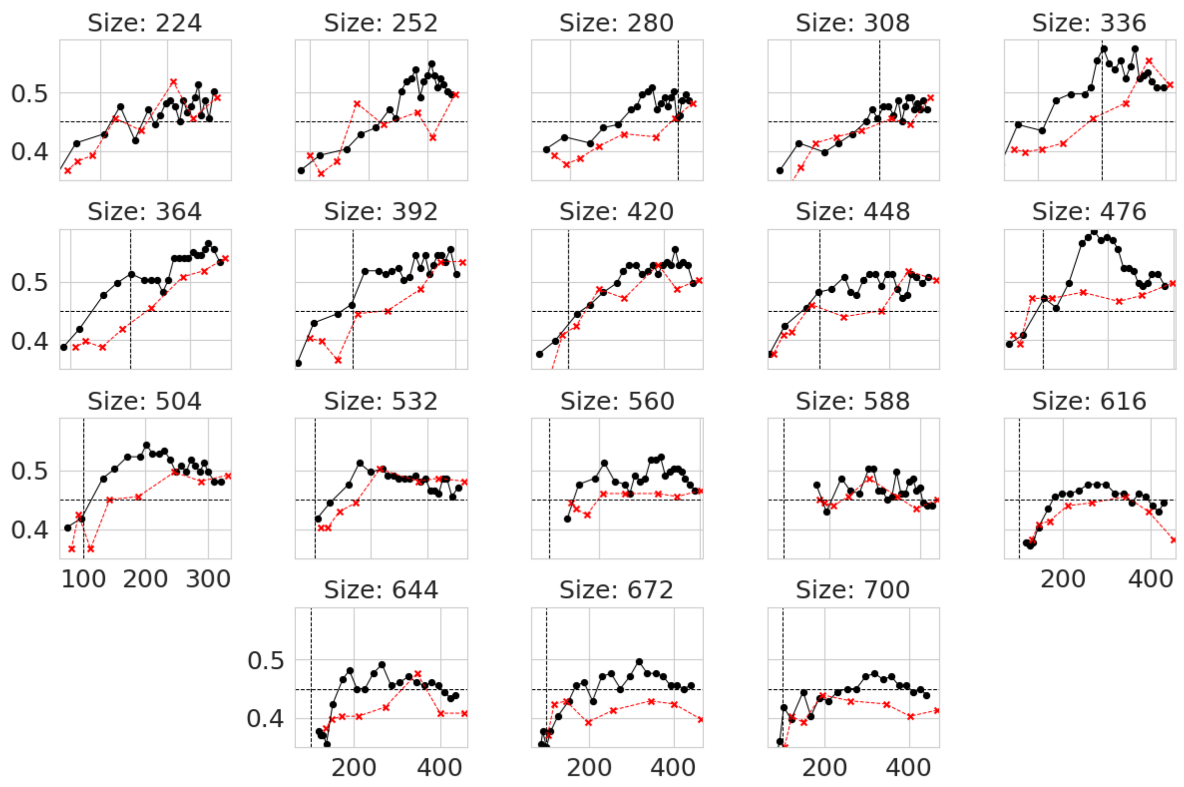


FIGURE 4.19. Ablation across a diverse range of image sizes of the QLIP-13B model on the V^* dataset. The Black line is the QLIP performance with the derivative selection strategy and the red line is a random selection strategy. Each random selection trial was only run once.

4.6. Related Work

Improved Vision Encoders and MLLMs The observation that grid patchification at a fixed image resolution is a poor inductive bias is not new. This has led to a litany of proposed replacements for CLIP [82] and ViT [34]. For example, the studies [12,28,31,36,41,49,57,59,70,73,79,113,117] propose modifications to the ViT architecture which provide better visual signal. These studies do not attempt to train an attendant LLM to create an MLLM. The studies [9,46,64,67,68,89,96,97,106,109,113,115] introduce new vision encoders specifically in the context of MLLM, but require pre-training and instruction tuning. The most closely related result work to ours is by Shi et al. [89] who show that LLaVA performance can be increased substantially by feeding the LLM visual tokens from different scales while keeping the CLIP encoder frozen. We go beyond all of these studies by obtaining improved performance using the *same* underlying MLLM backbone, with no pre-training, instruction-tuning, or supervised fine-tuning of the language model.

Token Pruning and Merging Many MLLM studies have been directed at reducing the number of visual input tokens, either by pruning tokens or merging them. Such reductions are well-motivated. [60,98]

show that in addition to being computationally expensive, feeding an LLM too many tokens can harm performance. Recent work has also demonstrated that MLLMs rely heavily on the `[CLS]` token during VQA [**116**], which helps explain why previous authors have been able to remove up to 95% of the visual tokens and nearly maintain MLLM performance [**18**, **19**, **52**, **93**, **95**], or prune tokens across video frames while maintaining performance [**23**]. However, all of these studies require an expensive pre-training and fine-tuning stage to align the LLM with their vision encoder. Furthermore, our work is orthogonal to the studies [**18**, **52**, **93**, **95**] since these models rely on training LLaVA family models while using the CLIP encoder, which can be replaced in their studies by QLIP.

4.7. Conclusion

We have proposed QLIP, a drop-in, adaptive, and content-aware replacement for the CLIP encoder. We defined mesoscopic bias and interpolation bias, argued that these biases are responsible for performance difficulties on fine-grained VQA, and shown that QLIP satisfactorily addresses these biases. We achieve +13.6% accuracy on the challenging V^* benchmark with *no fine-tuning or re-training* of the underlying MLLM. We are also able to exceed baseline performance on V^* while using fewer image tokens. On other benchmarks, we show that we can nearly match or exceed baseline performance.

Limitations

In Section 4.2 we assumed that the `[CLS]` token should be constant as a function of image size. This assumption, while stronger than the original implicit prior of CLIP, still lacks theoretical justification. It is easy to argue that a strong vision prior could be stated. The reason for this is that the CLIP encoder's understanding of an image may change as we scale image size, despite the theoretical alignment of the semantic content. For example, in the leftmost panel of Figure 4.2 we would not expect an image in which the elephant occupies 576 patches to have the same `[CLS]` embedding as the zoomed out version.

We did not fully sweep or optimize the MLP training due to compute limitations (c.f. Chapter 4.3.4 for a discussion of how we arrived at our hyper-parameters). Sweeping the MLP training hyper-parameters more fully would likely yield a better MLP model.

We trained the MLP on images that were smaller than or equal to 560 on their shortest edge. This was primarily an exercise in the tradeoff between batch-size and image-size. Training on larger images is preferable, but at the cost of smaller batch-sizes and significantly longer training times. We found that 560 was a happy medium for this trade-off. Future work could explore ways to train the MLP on very large images without actually loading the entirety of the image into memory. We believe such a methodology would be useful more broadly in the computer vision / multi-modal communities.

We also did not explore training the MLP on different datasets. While we believe that the content of the images is largely immaterial we suspect that the distribution of image sizes is quite important. We leave an investigation of this relationship to future work.

While we did explore multiple selection strategies (i.e. alternatives to (4.1), there is room for a more comprehensive and theoretically justified exploration of potential selection strategies. For example one could run a large-scale study correlating different selection methods with how well they find the ”correct” tokens predicted by [**61**].

Finally, we could run our model through a more comprehensive suite of benchmarks to gain a more accurate sense of its performance.

Bibliography

[1] D. Albritton and R. Beekie, *Long-time behavior of scalar conservation laws with critical dissipation*, arXiv, (2020).

[2] N. Alibaud, J. Droniou, and J. Vovelle, *Occurence and non-appearence of shocks in fractal burger's equation*, J. Hyperbolic Differ. Equ., 04 (2007), pp. 479–499.

[3] K. Antonion, X. Wang, M. Raissi, and L. Joshie, *Machine learning through physics–informed neural networks: Progress and challenges*, Academic Journal of Science and Technology, 9 (2024), pp. 46–49.

[4] G. Barenblatt, *Scaling, Self-similarity, and Intermediate Asymptotics: Dimensional Analysis and Intermediate Asymptotics*, Cambridge Texts in Applied Mathematics, Cambridge University Press, 1996.

[5] G. Barenblatt and Y. B. Zel'Dovich, *Self-similar solutions as intermediate asymptotics*, Annual Review of Fluid Mechanics, 4 (1972), pp. 285–312.

[6] G. I. Barenblatt and Y. B. Zel'dovich, *Self-similar solutions as intermediate asymptotics*, Annual Review of Fluid Mechanics, 4 (1972), pp. 285–312.

[7] A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, *Automatic differentiation in machine learning: a survey*, Journal of machine learning research, 18 (2018), pp. 1–43.

[8] A. Biasi, *Self-similar solutions to the compressible euler equations and their instabilities*, Communications in Nonlinear Science and Numerical Simulation, 103 (2021), p. 106014.

[9] M. Bigverdi, Z. Luo, C.-Y. Hsieh, E. Shen, D. Chen, L. G. Shapiro, and R. Krishna, *Perception tokens enhance visual reasoning in multimodal language models*, arXiv preprint arXiv:2412.03548, (2024).

[10] P. Biler, T. Funaki, and W. A. Woyczynski, *Fractal burgers equations*, Journal of Differential Equations, 148 (1998), pp. 9 – 46.

[11] R. Bischof and M. Kraus, *Multi-objective loss balancing for physics-informed deep learning*, arXiv preprint arXiv:2110.09813, (2021).

[12] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman, *Token merging: Your vit but faster*, arXiv preprint arXiv:2210.09461, (2022).

[13] R. A. Brualdi, *Introductory Combinatorics*, Pearson, 5 ed., 2010.

[14] T. Buckmaster and S. Iyer, *Formation of unstable shocks for 2d isentropic compressible euler*, arXiv, (2020).

[15] T. Buckmaster, S. Shkoller, and V. Vicol, *Formation of point shocks for 3d compressible euler*, arXiv, (2019).

[16] ———, *Shock formation and vorticity creation for 3d euler*, arXiv, (2019).

[17] ———, *Formation of shocks for 2d isentropic compressible euler*, Comm. Pure Appl. Math., to appear, (2020).

[18] Q. Cao, B. Paranjape, and H. Hajishirzi, *Pumer: Pruning and merging tokens for efficient vision language models*, arXiv preprint arXiv:2305.17530, (2023).

[19] L. Chen, H. Zhao, T. Liu, S. Bai, J. Lin, C. Zhou, and B. Chang, *An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models*, in European Conference on Computer Vision, Springer, 2024, pp. 19–35.

[20] K. R. Chickering, *A quasilinear algorithm for computing higher-order derivatives of deep feed-forward neural networks*, arXiv preprint arXiv:2412.09752, (2024).

[21] K. R. Chickering, B. Li, and M. Chen, *Qlip: A dynamic quadtree vision prior enhances mllm performance without retraining*, arXiv preprint arXiv:2505.23004, (2025).

[22] K. R. Chickering, R. C. Moreno-Vasquez, and G. Pandya, *Asymptotically self-similar shock formation for 1d fractal burgers' equation*, SIAM Journal on Mathematical Analysis, 55 (2023), pp. 7328–7360.

[23] R. Choudhury, G. Zhu, S. Liu, K. Niinuma, K. Kitani, and L. Jeni, *Don't look twice: Faster video transformers with run-length tokenization*, Advances in Neural Information Processing Systems, 37 (2024), pp. 28127–28149.

[24] C. Collot, T.-E. Ghoul, and N. Masmoudi, *Singularity formation for burgers equation with transverse viscosity*, arXiv, (2018).

[25] G. Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems, 2 (1989), pp. 303–314.

[26] W. M. Czarnecki, S. Osindero, M. Jaderberg, G. Swirszcz, and R. Pascanu, *Sobolev training for neural networks*, Advances in neural information processing systems, 30 (2017).

[27] A. Córdoba and D. Córdoba, *A maximum principle applied to quasi-geostrophic equations*, Commun. Math. Phys., 249 (2004), p. 511–528.

[28] T. Darcet, M. Oquab, J. Mairal, and P. Bojanowski, *Vision transformers need registers*, arXiv preprint arXiv:2309.16588, (2023).

[29] T. De Ryck, F. Bonnet, S. Mishra, and E. de Bézenac, *An operator preconditioning perspective on training in physics-informed machine learning*, arXiv preprint arXiv:2310.05801, (2023).

[30] T. De Ryck and S. Mishra, *Error analysis for physics-informed neural networks (pinns) approximating kolmogorov pdes*, Advances in Computational Mathematics, 48 (2022), p. 79.

[31] M. Dehghani, B. Mustafa, J. Djolonga, J. Heek, M. Minderer, M. Caron, A. Steiner, J. Puigcerver, R. Geirhos, I. M. Alabdulmohsin, et al., *Patch n'pack: Navit, a vision transformer for any aspect ratio and resolution*, Advances in Neural Information Processing Systems, 36 (2023), pp. 2252–2274.

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, *Imagenet: A large-scale hierarchical image database*, in 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[33] H. Dong, D. Du, and D. Li, *Finite time singularities and global well-posedness for fractal burgers equations*, Indiana University Mathematics Journal, 58 (2009), pp. 807–821.

[34] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al., *An image is worth 16x16 words: Transformers for image recognition at scale*, arXiv preprint arXiv:2010.11929, (2020).

[35] H. Duan, J. Yang, Y. Qiao, X. Fang, L. Chen, Y. Liu, X. Dong, Y. Zang, P. Zhang, J. Wang, et al., *Vlmevalkit: An open-source toolkit for evaluating large multi-modality models*, in Proceedings of the 32nd ACM International Conference on Multimedia, 2024, pp. 11198–11201.

[36] S. Duggal, P. Isola, A. Torralba, and W. T. Freeman, *Adaptive length image tokenization via recurrent allocation*, in First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models, 2024.

[37] J. Eggers and M. A. Fontelos, *The role of self-similarity in singularities of partial differential equations*, Nonlinearity, 22 (2008), pp. R1–R44.

[38] J. Eggers and M. A. Fontelos, *Singularities: Formation, Structure, and Propagation*, Cambridge Texts in Applied Mathematics, Cambridge University Press, 2015.

[39] J. Eggers and M. A. Fontelos, *Singularities: formation, structure, and propagation*, vol. 53, Cambridge University Press, 2015.

[40] L. C. Evans, *Partial differential equations*, vol. 19, American Mathematical Society, 2022.

[41] H. Fan, B. Xiong, K. Mangalam, Y. Li, Z. Yan, J. Malik, and C. Feichtenhofer, *Multiscale vision transformers*, in Proceedings of the IEEE/CVF international conference on computer vision, 2021, pp. 6824–6835.

[42] A. Farea, O. Yli-Harja, and F. Emmert-Streib, *Understanding physics-informed neural networks: Techniques, applications, trends, and challenges*, AI, 5 (2024), pp. 1534–1557.

[43] C. Fu, P. Chen, Y. Shen, Y. Qin, M. Zhang, X. Lin, J. Yang, X. Zheng, K. Li, X. Sun, et al., *Mme: A comprehensive evaluation benchmark for multimodal large language models*, arXiv preprint arXiv:2306.13394, (2023).

[44] Y. Giga and R. V. Kohn, *Asymptotically self-similar blow-up of semilinear heat equations*, Communications on Pure and Applied Mathematics, 38 (1985), pp. 297–319.

[45] Y. Giga and R. V. Kohn, *Characterizing blowup using similarity variables*, Indiana University Mathematics Journal, 36 (1987), pp. 1–40.

[46] Z. Guo, R. Xu, Y. Yao, J. Cui, Z. Ni, C. Ge, T.-S. Chua, Z. Liu, and G. Huang, *Llava-uhd: an lmm perceiving any aspect ratio and high-resolution images*, in European Conference on Computer Vision, Springer, 2024, pp. 390–406.

[47] R. Haberman, *Elementary applied partial differential equations: with Fourier series and boundary value problems*, Prentice-Hall Englewood Cliffs, NJ, 1987.

[48] G. H. Hardy and S. Ramanujan, *Asymptotic formulae for the distribution of integers of various types*, Proceedings of the London Mathematical Society, 2 (1917), pp. 112–132.

[49] J. B. Haurum, S. Escalera, G. W. Taylor, and T. B. Moeslund, *Which tokens to use? investigating token reduction in vision transformers*, in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2023, pp. 773–783.

[50] K. Hornik, *Approximation capabilities of multilayer feedforward networks*, Neural networks, 4 (1991), pp. 251–257.

[51] J. Howard, *Imagenette: A smaller subset of 10 easily classified classes from imagenet*, March 2019.

[52] W. Hu, Z.-Y. Dou, L. Li, A. Kamath, N. Peng, and K.-W. Chang, *Matryoshka query transformer for large vision-language models*, Advances in Neural Information Processing Systems, 37 (2024), pp. 50168–50188.

[53] G. M. Hunter and K. Steiglitz, *Operations on images using quad trees*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (1979), pp. 145–153.

[54] S. Ioffe and C. Szegedy, *Batch normalization: Accelerating deep network training by reducing internal covariate shift*, in International conference on machine learning, pmlr, 2015, pp. 448–456.

[55] D. P. Kingma, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, (2014).

[56] A. Kiselev, F. Nazarov, and R. Shterenberg, *Blow up and regularity for fractal burgers equation*, Dynamics of Partial Differential Equations, 5 (2008), p. 211–240.

[57] Z. Kong, P. Dong, X. Ma, X. Meng, W. Niu, M. Sun, X. Shen, G. Yuan, B. Ren, H. Tang, et al., *Spvit: Enabling faster vision transformers via latency-aware soft token pruning*, in European conference on computer vision, Springer, 2022, pp. 620–640.

[58] A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, *Characterizing possible failure modes in physics-informed neural networks*, Advances in neural information processing systems, 34 (2021), pp. 26548–26560.

[59] Y. Lee, J. Kim, J. Willette, and S. J. Hwang, *Mpvit: Multi-path vision transformer for dense prediction*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 7287–7296.

[60] M. Levy, A. Jacoby, and Y. Goldberg, *Same task, more tokens: the impact of input length on the reasoning performance of large language models*, arXiv preprint arXiv:2402.14848, (2024).

[61] B. Li, F. Wang, W. Zhou, N. Xu, and B. Zhou, *Semantic-clipping: Efficient vision-language modeling with semantic-guided visual selection*, arXiV.

[62] S. Li, P. W. Koh, and S. S. Du, *Exploring how generative mllms perceive more than clip with the same vision encoder*, https://arxiv.org/abs/2411.05195, (2024).

[63] Y. Li, Y. Du, K. Zhou, J. Wang, W. X. Zhao, and J.-R. Wen, *Evaluating object hallucination in large vision-language models*, arXiv preprint arXiv:2305.10355, (2023).

[64] H. Liu, C. Li, Y. Li, and Y. J. Lee, *Improved baselines with visual instruction tuning*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 26296–26306.

[65] H. Liu, C. Li, Q. Wu, and Y. J. Lee, *Visual instruction tuning*, Advances in neural information processing systems, 36 (2023), pp. 34892–34916.

[66] Y. Liu, H. Duan, Y. Zhang, B. Li, S. Zhang, W. Zhao, Y. Yuan, J. Wang, C. He, Z. Liu, et al., *Mmbench: Is your multi-modal model an all-around player?*, in European conference on computer vision, Springer, 2024, pp. 216–233.

[67] P. Lu, S. Mishra, T. Xia, L. Qiu, K.-W. Chang, S.-C. Zhu, O. Tafjord, P. Clark, and A. Kalyan, *Learn to explain: Multimodal reasoning via thought chains for science question answering*, in The 36th Conference on Neural Information Processing Systems (NeurIPS), 2022.

[68] G. Luo, Y. Zhou, Y. Zhang, X. Zheng, X. Sun, and R. Ji, *Feast your eyes: Mixture-of-resolution adaptation for multimodal large language models*, arXiv preprint arXiv:2403.03003, (2024).

[69] S. Maddu, D. Sturm, C. L. Müller, and I. F. Sbalzarini, *Inverse dirichlet weighting enables reliable training of physics informed neural networks*, Machine Learning: Science and Technology, 3 (2022), p. 015026.

[70] D. Marin, J.-H. R. Chang, A. Ranjan, A. Prabhu, M. Rastegari, and O. Tuzel, *Token pooling in vision transformers for image classification*, in Proceedings of the IEEE/CVF winter conference on applications of computer vision, 2023, pp. 12–21.

[71] S. Markidis, *The old and the new: Can physics-informed deep-learning replace traditional linear solvers?*, Frontiers in big Data, 4 (2021), p. 669097.

[72] N. McGreivy and A. Hakim, *Weak baselines and reporting biases lead to overoptimism in machine learning for fluid-related partial differential equations*, Nature Machine Intelligence, (2024), pp. 1–14.

[73] L. Meng, H. Li, B.-C. Chen, S. Lan, Z. Wu, Y.-G. Jiang, and S.-N. Lim, *Adavit: Adaptive vision transformers for efficient image recognition*, in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 12309–12318.

[74] S. Mishra and R. Molinaro, *Estimates on the generalization error of physics-informed neural networks for approximating pdes*, IMA Journal of Numerical Analysis, 43 (2023), pp. 1–43.

[75] J. Ng, Y. Wang, and C.-Y. Lai, *Spectrum-informed multistage neural networks: Multiscale function approximators of machine precision*, arXiv preprint arXiv:2407.17213, (2024).

[76] J. Nocedal and S. J. Wright, *Numerical optimization*, Springer, 1999.

[77] S.-J. Oh and F. Pasqualotto, *Gradient blow-up for dispersive and dissipative perturbations of burgers equation*, In Preperation, (2021).

[78] S.-J. Oh and F. Pasqualotto, *Gradient blow-up for dispersive and dissipative perturbations of the burgers equation*, Archive for Rational Mechanics and Analysis, 248 (2024), p. 54.

[79] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al., *Dinov2: Learning robust visual features without supervision*, arXiv preprint arXiv:2304.07193, (2023).

[80] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al., *Pytorch: An imperative style, high-performance deep learning library*, Advances in neural information processing systems, 32 (2019).

[81] Y. Qiu and L. Zhao, *Formation of multi-point singularities of self-similar type for burgers equation*, arXiv, (2021).

[82] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al., *Learning transferable visual models from natural language supervision*, in International conference on machine learning, PmLR, 2021, pp. 8748–8763.

[83] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, and A. Courville, *On the spectral bias of neural networks*, in International conference on machine learning, PMLR, 2019, pp. 5301–5310.

[84] M. Raissi, P. Perdikaris, and G. E. Karniadakis, *Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations*, Journal of Computational physics, 378 (2019), pp. 686–707.

[85] E. Ramírez and B. Protas, *Singularity formation in the deterministic and stochastic fractional burgers equation*, arXiv, (2021).

[86] P. Rathore, W. Lei, Z. Frangella, L. Lu, and M. Udell, *Challenges in training pinns: A loss landscape perspective*, arXiv preprint arXiv:2402.01868, (2024).

[87] S. Roman, *The formula of faa di bruno*, The American Mathematical Monthly, 87 (1980), pp. 805–809.

[88] A. V. S. Akopian, M. J. Kang, *Inviscid limit to the shock waves for the fractal burgers equation*, 2020.

[89] B. Shi, Z. Wu, M. Mao, X. Wang, and T. Darrell, *When do we not need larger vision models?*, in European Conference on Computer Vision, Springer, 2024, pp. 444–462.

[90] H.-J. M. Shi and D. Mudigere, *Pytorch l-bfgs*. https://github.com/hjmshi/PyTorch-LBFGS, 2020.

[91] P. Simard, B. Victorri, Y. LeCun, and J. Denker, *Tangent prop-a formalism for specifying selected invariances in an adaptive network*, Advances in neural information processing systems, 4 (1991).

[92] H. Son, J. W. Jang, W. J. Han, and H. J. Hwang, *Sobolev training for physics informed neural networks*, arXiv preprint arXiv:2101.08932, (2021).

[93] Y. Sun, Y. Xin, H. Li, J. Sun, C. Lin, and R. Batista-Navarro, *Lvpruning: An effective yet simple language-guided vision token pruning approach for multi-modal large language models*, arXiv preprint arXiv:2501.13652, (2025).

[94] M. Tancik, P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. Barron, and R. Ng, *Fourier features let networks learn high frequency functions in low dimensional domains*, Advances in neural information processing systems, 33 (2020), pp. 7537–7547.

[95] S. Tang, J. Zhang, S. Zhu, and P. Tan, *Quadtree attention for vision transformers*, arXiv preprint arXiv:2201.02767, (2022).

[96] R. Thapa, K. Chen, I. Covert, R. Chalamala, B. Athiwaratkun, S. L. Song, and J. Zou, *Dragonfly: Multi-resolution zoom-in encoding enhances vision-language models*, arXiv preprint arXiv:2406.00977, (2024).

[97] P. Tong, E. Brown, P. Wu, S. Woo, A. J. V. IYER, S. C. Akula, S. Yang, J. Yang, M. Middepogu, Z. Wang, et al., *Cambrian-1: A fully open, vision-centric exploration of multimodal llms*, Advances in Neural Information Processing Systems, 37 (2024), pp. 87310–87356.

[98] P. Veličković, C. Perivolaropoulos, F. Barbero, and R. Pascanu, *softmax is not enough (for sharp out-of-distribution)*, arXiv preprint arXiv:2410.01104, (2024).

[99] G. K. Wallace, *The jpeg still picture compression standard*, IEEE transactions on consumer electronics, 38 (1992), pp. xviii–xxxiv.

[100] H. Wang, Y. Cao, Z. Huang, Y. Liu, P. Hu, X. Luo, Z. Song, W. Zhao, J. Liu, J. Sun, et al., *Recent advances on machine learning for computational fluid dynamics: A survey*, arXiv preprint arXiv:2408.12171, (2024).

[101] S. Wang, S. Sankaran, H. Wang, and P. Perdikaris, *An expert's guide to training physics-informed neural networks*, arXiv preprint arXiv:2308.08468, (2023).

[102] S. Wang, Y. Teng, and P. Perdikaris, *Understanding and mitigating gradient pathologies in physics-informed neural networks*, arXiv preprint arXiv:2001.04536, (2020).

[103] S. Wang, X. Yu, and P. Perdikaris, *When and why pinns fail to train: A neural tangent kernel perspective*, Journal of Computational Physics, 449 (2022), p. 110768.

[104] Y. Wang and C.-Y. Lai, *Multi-stage neural networks: Function approximator of machine precision*, Journal of Computational Physics, 504 (2024), p. 112865.

[105] Y. Wang, C.-Y. Lai, J. Gómez-Serrano, and T. Buckmaster, *Asymptotic self-similar blow-up profile for three-dimensional axisymmetric euler equations using neural networks*, Physical Review Letters, 130 (2023), p. 244002.

[106] Y. Wang, W. Zhou, H. Feng, and H. Li, *Adaptvision: Dynamic input scaling in mllms for versatile scene understanding*, arXiv preprint arXiv:2408.16986, (2024).

[107] G. Whitham, *Linear and Nonlinear Waves*, Pure and Applied Mathematics: A Wiley Series of Texts, Monographs and Tracts, Wiley, 2011.

[108] P. Wu and S. Xie, *V?: Guided visual search as a core mechanism in multimodal llms*, in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024, pp. 13084–13094.

[109] S. Wu, H. Fei, X. Li, J. Ji, H. Zhang, T.-S. Chua, and S. Yan, *Towards semantic equivalence of tokenization in multimodal llm*, arXiv preprint arXiv:2406.05127, (2024).

[110] xAI Team, *Grok-1.5 vision preview*, 2024.

[111] Z.-Q. J. Xu, Y. Zhang, T. Luo, Y. Xiao, and Z. Ma, *Frequency principle: Fourier analysis sheds light on deep neural networks*, arXiv preprint arXiv:1901.06523, (2019).

[112] R. Yang, *Shock formation for the burgers-hilbert equation*, arXiv, (2020).

[113] T. Yang, Y. Wang, Y. Lu, and N. Zheng, *Visual concepts tokenization*, Advances in Neural Information Processing Systems, 35 (2022), pp. 31571–31582.

[114] Y. Yang and J. He, *Deeper or wider: A perspective from optimal generalization error with sobolev loss*, arXiv preprint arXiv:2402.00152, (2024).

[115] J. Zhang, W. Yang, S. Lai, Z. Xie, and L. Jin, *Dockylin: A large multimodal model for visual document understanding with efficient visual slimming*, in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 39, 2025, pp. 9923–9932.

[116] Q. Zhang, A. Cheng, M. Lu, Z. Zhuo, M. Wang, J. Cao, S. Guo, Q. She, and S. Zhang, *[cls] attention is all you need for training-free visual token pruning: Make vlm inference faster*, arXiv preprint arXiv:2412.01818, (2024).

[117] Q. Zhang, Y. Xu, J. Zhang, and D. Tao, *Vsa: Learning varied-size window attention in vision transformers*, in European conference on computer vision, Springer, 2022, pp. 466–483.