Generalization Bounds for Solving Navier-Stokes PDE by Neural Nets

Dibyakanti Kumar, Department of Computer Science, The University of Manchester

Collaborators

This work is with,

Sébastien André-Sloan @ UManchester, (PhD Candidate, CSE),
Alejandro Frangi @ UManchester (Computer Science)
&
Anirbit Mukherjee @ UManchester (Computer Science)

Physics Informed Neural Networks

Physics Informed Neural Nets (PINNs)

Let $u(x, t)$ be the actual solution

PDE

$$u_t + \mathcal{N}_x[u] = 0, \quad x \in D, t \in [0, T]$$

Initial Condition

$$u(x, 0) = h(x), \quad x \in D$$

Boundary Conditions

$$u(x, t) = g(x, t), \quad t \in [0, T], x \in \partial D$$

Physics Informed Neural Nets (PINNs)

Let the neural solution be $\boldsymbol{u}_\theta(\boldsymbol{x}, t)$

Functional Residual

$$\mathcal{R}_{pde,\theta}(x, t) \coloneqq \partial_t \boldsymbol{u}_\theta + \mathcal{N}_x[\boldsymbol{u}_\theta(\boldsymbol{x}, t)]$$

Conditional Residual (I.C.)

$$\mathcal{R}_{t,\theta}(x) \coloneqq \boldsymbol{u}_\theta(\boldsymbol{x}, 0) - h(\boldsymbol{x})$$

Conditional Residual (B.C.)

$$\mathcal{R}_{b,\theta}(x, t) \coloneqq \boldsymbol{u}_\theta(\boldsymbol{x}, t) - g(\boldsymbol{x}, t)$$

Let the neural solution be $\boldsymbol{u}_\theta(\boldsymbol{x}, t)$

Functional Residual

$$\mathcal{R}_{pde,\theta}(x,t) \coloneqq \partial_t \boldsymbol{u}_\theta + \mathcal{N}_x[\boldsymbol{u}_\theta(\boldsymbol{x},t)]$$

Conditional Residual (I.C.)

$$\mathcal{R}_{t,\theta}(x) \coloneqq \boldsymbol{u}_\theta(\boldsymbol{x},0) - h(\boldsymbol{x})$$

Conditional Residual (B.C.)

$$\mathcal{R}_{b,\theta}(x,t) \coloneqq \boldsymbol{u}_\theta(\boldsymbol{x},t) - g(\boldsymbol{x},t)$$

PINN Loss Function

$$\mathcal{L}_{\mathcal{S}}(\theta) \coloneqq \frac{1}{N_{pde}} \sum_{i=1}^{N_{pde}} \mathcal{R}_{pde,\theta}(x_r^i, t_r^i)^2 + \lambda_1 \frac{1}{N_t} \sum_{i=1}^{N_t} \mathcal{R}_{t,\theta}(x_t^i)^2 + \lambda_2 \frac{1}{N_b} \sum_{i=1}^{N_b} \mathcal{R}_{b,\theta}(x_b^i, t_b^i)^2$$

Why Physics Informed Neural Nets (PINNs) ?

$$\mathcal{E}_G := \|u - u_\theta\|$$

$$\tilde{\mathcal{E}}_T := \left(\sum_{n=1}^{N} w_n |\mathcal{R}_\theta|^p \right)^{1/p}$$

Risk Upperbound (Theorem 2.6)[1]

$$\mathcal{E}_G \leq C_{pde} \tilde{\mathcal{E}}_T + C_{pde} C_{quad}^{\frac{1}{p}} N^{-\frac{\alpha}{p}}$$

It was proven in the above mentioned theorem – for the first time – that one can minimize certain empirical errors to find provably good approximations to any PDE.

[1] Mishra and Molinaro, "Estimates on the generalization error of physics-informed neural networks for approximating PDEs".

Why Physics Informed Neural Nets (PINNs) ?

Upperbounds of L2-risk and Residual for any PDE (Theorem 3.5)[2]

$$\|\boldsymbol{u} - \boldsymbol{u}_\theta\|_{L^2} + \left\|R_{pde,\theta}\right\|_{L^2} \leq C_1 \cdot \text{poly}(d) \cdot C_u$$

- This showed that there **exists** neural networks that can simultaneously reduce the PDE loss and the L2-risk.
- **Does not provide a finite-sample bound.**

[2] De Ryck and Mishra, "Generic bounds on the approximation error for physics-informed (and) operator learning".

Why Physics Informed Neural Nets (PINNs) ?

Upperbounds of Risk for Linear PDE (Theorem 2)[3]

$$\|\boldsymbol{u} - \boldsymbol{u}_\theta\|_{H^s} \leq \sqrt{C_u(\mathbb{E}[\boldsymbol{u}_\theta] - \inf_\psi \mathbb{E}[\boldsymbol{u}_\psi]) + C_{pde} \inf_\psi \|\boldsymbol{u}_\psi - \boldsymbol{u}\|}$$

- This showed that there **exists** neural networks that can reduce the risk in Sobolev norm for PINNs.
- **Does not provide a finite-sample bound.**

[3] Zeinhofer, Masri, and Mardal, "A unified framework for the error analysis of physics-informed neural networks".

Why Physics Informed Neural Nets (PINNs) ?

Upperbounds of L2-risk for NS (Theorem 3.4)[4]

$$\|\boldsymbol{u} - \boldsymbol{u}_\theta\|_{L^2} \leq C_2 \cdot C_{\nabla u}$$

- This showed that there **exists** neural networks that can reduce the L2-risk for PINNs when solved for Navier-Stokes.
- Does not provide a finite-sample bound.

[4] De Ryck, Jagtap, and Mishra, "Error estimates for physics-informed neural networks approximating the Navier–Stokes equations".

Why PINNs for Navier–Stokes?

Recent studies have demonstrated the effectiveness of solving fluid flow problems with a physics-informed loss.

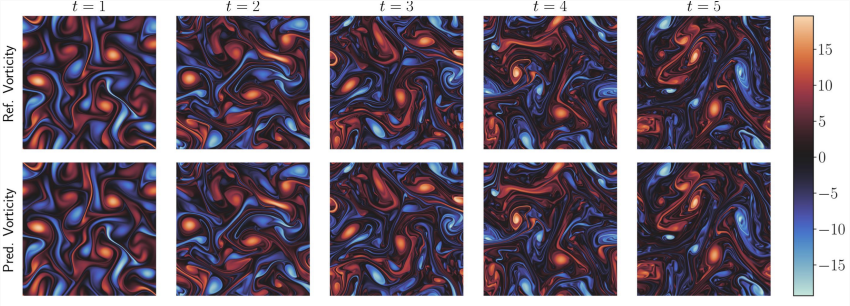

Figure 1: Time evolution of the 2D Kolmogorov flow solved using PirateNet, as reported in a recent study[5].

[5] Wang et al., "Simulating three-dimensional turbulence with physics-informed neural networks".

What are We Missing?

But “learning” is far more than just an approximation guarantee!

We need to explain why the PINNs gives good approximations at **points that never occurred in the training data.**

What are We Missing?

But “learning” is far more than just an approximation guarantee!

We need to explain why the PINNs gives good approximations at **points that never occurred in the training data.**

This brings us to the core question in learning theory
- that of quantifying **“generalization error”**.

Rademacher Complexity

What Makes Up a Deep Learning Model?

The following choices must be made with care:

1. Network architecture (e.g., size, activation functions, ...)
2. Training algorithm
3. Data
4. Loss Function

What Makes Up a Deep Learning Model?

The following choices must be made with care:

1. Network architecture (e.g., size, activation functions, ...)
2. Training algorithm
3. Data
4. Loss Function

What can we expect to achieve in theory?

- The **optimal combination** of these four components
- A **provably good** choice of these components

What Makes Up a Deep Learning Model?

The following choices must be made with care:

1. Network architecture (e.g., size, activation functions, ...)
2. Training algorithm
3. Data
4. Loss Function

What can we expect to achieve in theory?

- The **optimal combination** of these four components
- A **provably good** choice of these components ← This is where **Rademacher complexity** can guide us.

What is Rademacher Complexity?

Suppose being given a function class $\mathcal{H}$ mapping into $\mathbb{R}$, $\ell(\cdot)$ be a $L-$Lipschitz loss function and a $m-$sized data-set $\mathcal{D}$ of points $\{x_i \mid i = 1, \dots, m\}$ in $\mathrm{domain}\,(\mathcal{H})$.

Then, for $\epsilon_i \sim \mathrm{Uniform}(\pm 1)$, the **empirical Rademacher complexity** is,

$$\hat{\mathcal{R}}_m(\ell(\mathcal{H})) = \mathbb{E}_\epsilon \left[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \epsilon_i \ell(h(x_i)) \right].$$

If the data $\mathcal{D}$ above are sampled from a distribution P, then the **average Rademacher complexity** is,

$$\mathcal{R}_m(\ell(\mathcal{H})) = \mathbb{E}_{x_1,\dots,x_m \sim P} \left(\mathbb{E}_\epsilon \left[\sup_{h \in \mathcal{H}} \frac{1}{m} \sum_{i=1}^{m} \epsilon_i \ell(h(x_i)) \right] \right).$$

Is Rademacher Inevitable?

Don’t worry — the irony isn’t lost on me.

Is Rademacher Inevitable?

Theorem

$$\mathbb{E}_{x_1,\ldots,x_m\sim P}\left[\sup_{h\in\mathcal{H}}\left(\frac{1}{m}\cdot\sum_{i=1}^{m}\ell(h(x_i))-\mathbb{E}_{x\sim P}[\ell(h(x))]\right)\right]\leq 2\cdot \mathrm{Rad}_m(\ell(\mathcal{H}))$$

- In real-world uses ℓ is the loss function of the predictor h – and hence the LHS measures the "data averaged worst gap possible between the empirical loss and the true risk of the predictor".

Is Rademacher Inevitable?

Theorem

$$\mathbb{E}_{x_1,\ldots,x_m\sim P}\left[\sup_{h\in\mathcal{H}}\left(\frac{1}{m}\cdot\sum_{i=1}^{m}\ell(h(x_i))-\mathbb{E}_{x\sim P}[\ell(h(x))]\right)\right]\leq 2\cdot \mathrm{Rad}_m(\ell(\mathcal{H}))$$

- In real-world uses ℓ is the loss function of the predictor h – and hence the LHS measures the "data averaged worst gap possible between the empirical loss and the true risk of the predictor".
- This philosophically says that the ability of the predictor class to replicate uniform binary noise on a discrete set (its "capacity") has to be small for it to have the ability to "learn".

Gen. Error Bound for Linear 2nd-Order PDEs (Theorem 3.2)[6]

$$\mathbb{E}\left[\sup_{\theta}(\mathcal{L}_{\mathcal{S}}(\theta) - \mathbb{E}_{\mathcal{S}}\mathcal{L}_{\mathcal{S}}(\theta))\right] \leq \frac{C_1}{N_r\sqrt{N_r}} + \frac{C_2 \log N_r}{\sqrt{N_r}} + \frac{C_3}{N_b\sqrt{N_b}} + \frac{C_4 \log N_b}{\sqrt{N_b}}$$

- This is a finite-sample complexity bound to the generalization error.

[6] Hu et al., "When Do Extended Physics-Informed Neural Networks (XPINNs) Improve Generalization?"

Rademacher Complexity for Navier-Stokes

Rademacher Complexity for (d+1)-Navier-Stokes

Let the output of the neural net we aim to train be $\boldsymbol{u}_\theta(\boldsymbol{x}, t) \in \mathbb{R}^d$ and $p_\theta(\boldsymbol{x}, t) \in \mathbb{R}$.

Then the residual terms for this predictor are,

Conditional Residual

$$\mathcal{R}_{\mathrm{t},\theta} \coloneqq \boldsymbol{u}_\theta(t = t_0) - \boldsymbol{u}(t = t_0)$$

The residual term for the functional loss is

Functional Residual

$$\mathcal{R}_{\mathrm{pde},\theta} \coloneqq \partial_t \boldsymbol{u}_\theta + (\boldsymbol{u}_\theta \cdot \nabla)\boldsymbol{u}_\theta + \nabla p - \nu \nabla^2 \boldsymbol{u}_\theta; \quad \mathcal{R}_{\mathrm{div},\theta} \coloneqq \nabla \cdot \boldsymbol{u}_\theta$$

Rademacher Complexity for (d+1)-Navier-Stokes

Generalization Error

$$\mathcal{E}_{gen,\theta} := \frac{1}{N_r}\sum_{i=1}^{N_r}\mathcal{R}_{pde,\theta}(\boldsymbol{x}_{ri}, t_{ri}) + \frac{\lambda_0}{N_r}\sum_{i=1}^{N_r}\mathcal{R}_{div,\theta}(\boldsymbol{x}_{ri}, t_{ri}) + \frac{\lambda_1}{N_0}\sum_{j=1}^{N_0}\mathcal{R}_{t,\theta}(x_{0j})$$
$$- \left(\mathbb{E}_{(\boldsymbol{x}_r,t_r)\sim\mathcal{D}_1}[\mathcal{R}_{pde,\theta}(\boldsymbol{x}_r, t_r) + \lambda_0\mathcal{R}_{div,\theta}(\boldsymbol{x}_r, t_r)] + \lambda_1\mathbb{E}_{\boldsymbol{x}_0\sim\mathcal{D}_2}[R_{t,\theta}(\boldsymbol{x}_0)]\right)$$

Generalization Error

$$\mathcal{E}_{gen,\theta} := \frac{1}{N_r}\sum_{i=1}^{N_r}\mathcal{R}_{pde,\theta}(\boldsymbol{x}_{ri}, t_{ri}) + \frac{\lambda_0}{N_r}\sum_{i=1}^{N_r}\mathcal{R}_{div,\theta}(\boldsymbol{x}_{ri}, t_{ri}) + \frac{\lambda_1}{N_0}\sum_{j=1}^{N_0}\mathcal{R}_{t,\theta}(x_{0j})$$
$$- \left(\mathbb{E}_{(\boldsymbol{x}_r,t_r)\sim\mathcal{D}_1}[\mathcal{R}_{pde,\theta}(\boldsymbol{x}_r, t_r) + \lambda_0\mathcal{R}_{div,\theta}(\boldsymbol{x}_r, t_r)] + \lambda_1\mathbb{E}_{\boldsymbol{x}_0\sim\mathcal{D}_2}[R_{t,\theta}(\boldsymbol{x}_0)]\right)$$

Theorem 1

The worst case data-averaged generalization error can be bounded as,

$$\mathbb{E}\left[\sup_{\theta}\left(\mathcal{E}_{gen,\theta}\right)\right] \leq \frac{C_r}{\sqrt{N_r}} + \frac{C_0}{\sqrt{N_0}}.$$

Rademacher Complexity for (d+1)-Navier-Stokes

Theorem 1

The worst case data-averaged generalization error can be bounded as,

$$\mathbb{E}\left[\sup_{\theta}\left(\mathcal{E}_{gen,\theta}\right)\right] \leq \frac{C_r}{\sqrt{N_r}} + \frac{C_0}{\sqrt{N_0}}.$$

- The loss is Lipschitz (i.e. Huber loss)
- Depth-2 neural nets
- tanh activation

Rademacher Complexity for (d+1)-Navier-Stokes

Theorem 1

The worst case data-averaged generalization error can be bounded as,

$$\mathbb{E}\left[\sup_{\theta}\left(\mathcal{E}_{gen,\theta}\right)\right] \leq \frac{C_r}{\sqrt{N_r}} + \frac{C_0}{\sqrt{N_0}}.$$

- The loss is Lipschitz (i.e. Huber loss)
- Depth-2 neural nets
- tanh activation

This is possible the first generalization error bound for solving a non-linear PDE by neural nets.

An Overview of the Main Lemmas

Lipschitz Composition in Rademacher Complexity

Talagrand's Contraction Lemma

If ℓ is an L_ℓ-Lipschitz loss function, then

$$\mathcal{R}_m(\ell(\mathcal{H})) \leq L_\ell \cdot \mathcal{R}_m(\mathcal{H})$$

Why Do We Need a Contraction Lemma?

The key idea behind the proof is to establish a suitable **contraction lemma** for each component of the loss function.

After applying these contraction lemmas, we aim to reduce the problem to analyzing the **Rademacher complexity of a linear model**.

Why Do We Need a Contraction Lemma?

The key idea behind the proof is to establish a suitable **contraction lemma** for each component of the loss function.

After applying these contraction lemmas, we aim to reduce the problem to analyzing the **Rademacher complexity of a linear model**.

In our case, we developed **two contraction lemmas**: one for the non-linear term, and one for the remaining terms.

Contraction Lemma I

- $\phi : \mathbb{R} \to \mathbb{R}$ is L_ϕ-Lipschitz
- For any $\boldsymbol{\epsilon} \in \{\pm 1\}^N$, $\sup_{\boldsymbol{w}} \left\langle \boldsymbol{\epsilon}, \phi(\boldsymbol{Z}^\top \boldsymbol{w}) \right\rangle \geq 0$, where $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N]$
- $\sum_{m=1}^{p} |f(\boldsymbol{W})_m| \leq B$, where $f(\boldsymbol{W}) : \mathbb{R}^{p \times d} \to \mathbb{R}^p$

Contraction Lemma I

$$\mathbb{E}_{\boldsymbol{\epsilon} \sim \{\pm 1\}^N} \left[\sup_{\boldsymbol{W}} \left(\frac{1}{N} \sum_{i=1}^{N} \epsilon_i \left\langle f(\boldsymbol{W}), \phi(\boldsymbol{W} \boldsymbol{z}_i) \right\rangle \right) \right] \leq \frac{2BL_\phi}{N} \mathbb{E}_{\boldsymbol{\epsilon} \sim \{\pm 1\}^N} \left[\sup_{\bar{\boldsymbol{w}}} \sum_{i=1}^{N} \epsilon_i \left\langle \bar{\boldsymbol{w}}, \boldsymbol{z}_i \right\rangle \right]$$

Contraction Lemma II : Non-Linear Term

- $\phi_1, \phi_2 : \mathbb{R} \to \mathbb{R}$ are L_{ϕ_1}, L_{ϕ_2}-Lipschitz, and $\phi_1 \leq B_{\phi_1}$ and $\phi_2 \leq B_{\phi_2}$
- $\phi_1(\cdot)\phi_2(\cdot)$ is an odd function
- $f_m(\boldsymbol{W}) : \mathbb{R}^{p \times d} \to \mathbb{R}^p$ such that $\forall m \in [d], \sum_{q_1, q_2 = 1}^{p} \left| f_m(\boldsymbol{W})_{q_1} \right| \left| \boldsymbol{a}_{q_2} \right| \leq B$

Contraction Lemma II

$$\mathbb{E}_{\boldsymbol{\epsilon} \sim \{\pm 1\}^N} \left[\sup_{\boldsymbol{W}} \left(\frac{1}{N} \sum_{i=1}^{N} \sum_{m=1}^{d} \epsilon_i \left\langle f_m(\boldsymbol{W}), \phi_1(\boldsymbol{W} \boldsymbol{z}_i) \right\rangle \left\langle \boldsymbol{a}_m, \phi_2(\boldsymbol{W} \boldsymbol{z}_i) \right\rangle \right) \right]$$
$$\leq \frac{2B(B_{\phi_1} L_{\phi_2} + B_{\phi_2} L_{\phi_1})}{N} \mathbb{E}_{\boldsymbol{\epsilon} \sim \{\pm 1\}^N} \left[\sup_{\bar{\boldsymbol{w}}} \left(\sum_{i=1}^{N} \epsilon_i \left\langle \bar{\boldsymbol{w}}, \boldsymbol{z}_i \right\rangle \right) \right]$$

Putting It All Together

$$\mathbb{E}\left[\sup_{\theta}\left(\mathcal{E}_{gen,\theta}\right)\right] \leq \frac{C_r}{\sqrt{N_r}} + \frac{C_0}{\sqrt{N_0}}.$$

$$C_r := L_\ell B_w C_z (2B_{f_1} L_{\sigma'} + 2B_{f_2}(B_\sigma L_{\sigma'} + B_{\sigma'} L_\sigma) + 2B_{f_3} L_{\sigma'} + 4\nu B_{f_4}(L_{\sigma'} + L_\sigma) + 2\lambda_0 B_{f_5} L_{\sigma'})$$

$$C_0 := 2\lambda_1 L_\ell B_w C_{x_0} L_\sigma B_a$$

Looking Ahead

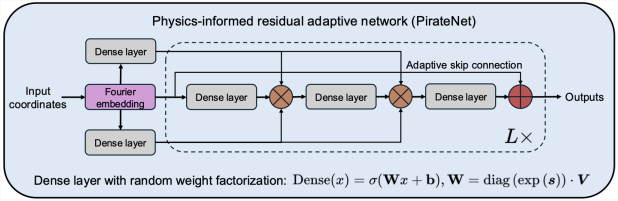

Figure 2: PirateNet architecture, as used in a recent study[7].

SOTA architectures like PirateNet are complex, and more theory is needed to explain their success on Navier–Stokes.

Much work remains to move from provably good settings to truly optimal ones that also consider loss optimizability.

[7] Wang et al., "Simulating three-dimensional turbulence with physics-informed neural networks".

Thank You!

To learn more about our work, scan the QR code or visit:

`https://dibyakanti.github.io/publication/`