Can the PDE Transformer learn complex physical laws?

by Paul Thompson, 7/2/2025

I suspect this remarkable paper [1] on the PDE-Transformer\* will help vision-language models (VLMs) and vision-language-action models (VLAs) to model the external world much better, especially when trained on real-time and offline video.

Remember that one criticism of large-language models as a means to solving very general problems (AGI/artificial general intelligence) is based on their limited "world model" if only language is used; but this type of method should lead to highly compact (low-MDL) world models that AI methods can use to interpret inputs, think, simulate, and re-think...

\*and related work on neural operators e.g. by Anima Anandkumar.

OK a short thread on transformers learning complex physical laws.

I am trying to see if yesterday's PDE-transformer could learn to solve a subordination problem (where one physical law is superimposed on another), without knowing the physical laws involved (e.g. Gray-Scott reactions, advection of smoke, advection of probability density/Fokker-Planck) To solve PDEs, they use multi‑channel attention.

In the context of Navier–Stokes, variables such as velocity components (eg u, v) and pressure/density are treated as separate channels and embedded into tokens; in the Separate‑Channel (SC) variant, each physical variable is embedded into its own set of tokens.

Those tokens are then arranged spatially and fed into local-window attention over space and \*channel‑axis attention\* is used to let information pass between variables so it looks like there are 2 types of attention going on: (1) the usual spatial type within a channel, i.e. you can apply a Laplacian filter to a variable like velocity, Tokens representing, say, the u‑velocity at neighboring grid patches attend to each other. This allows the model to infer numerical stencils, like approximating ∂u/∂x or ∂²u/∂y², purely from data.

It does not look like they use position encoding (need to check). you could use to enforce boundary conditions like Dirichlet/Neumann but it seems like they just set the variables to satisfy these at each iteration, which is what people usually do.

They also implement Channel Attention (Different Channels), where tokens from different variables-u, v, pressure-become “aware” of one another via attention along the channel dimension. So, a token for u can directly attend to the token for v and pressure at the same spatial patch, so the model gets to capture cross-variable couplings like the advection term u·∇v in Navier–Stokes.

This channel‑axis attention makes sure that the model learns how different physical quantities interact in the PDE dynamics

💡so when it is learning to solve Navier-Stokes, when wthin a token's local window, spatial attention learns local derivatives, the core idea behind a stencil. But between channels at each spatial location, channel attention models the physics of how pressure gradients drive velocity, how velocities alter momentum.

💡it is not clear to me whether stacking these attention operations across transformer layers approximates iterative time updates, aike stepping the PDE forward in time ?

💡anyway, the 2 types of attention (within and across channel) enable the model to emulate a data-driven PDE solver, automatically discovering stencils and update schemes from observations alone.

I can't tell if my understanding is 100% correct but it is interesting.

💡I am trying to see if it could learn to solve a subordination (where one PDE is superimposed on another), without knowing the physical law (e.g. Gray Scott superomposed on Navier Stokes). i.e. if it does not “know” there are two separate PDE processes

this is a test of pure data-driven discovery of intertwined processes. It would have to:

1. Learn that ∂t v is governed by velocity + pressure interactions (Navier–Stokes).

2. Learn that ∂t u involves local Laplacian (diffusion), local reaction, and also is transported by v.

and you could see what it is doing if you look inside attention maps and ask

1. Does the attention on u,w tokens query velocity tokens? (so it learned advection.)

2. Does velocity mostly attend to velocity + pressure? (so it learned Navier-Stokes dynamics.)

if it can do this, I suspect that it can probably do well in genetics and omics as well, where biological processes are subordinated and there is no direct law across all processes.

A red and blue circular object

Description automatically generatedA screenshot of a computer generated image

Description automatically generated

# References

[1] [PDE-Transformer: Efficient and Versatile Transformers for Physics Simulations, Benjamin Holzschuh, Qiang Liu, Georg Kohl, Nils Thuerey, 2025](https://github.com/dimitarpg13/deep_learning_for_dynamical_systems/blob/main/articles/PDE-Transformer-Efficient_and_Versatile_Transformers_for_Physics_Simulations_Holzschuh_2025.pdf)