SciML 2: Neural operators

E. Franck

Master CSMI, Strasbourg University

01/09/2025

1. Operator learning: rappel

Rappel I

Definition (Nonlocal parametric operator): On se donne une fonction $\mathbf{v}_l : \mathbb{R}^d \to \mathbb{R}^{in}$ et $\mathbf{v}_{l+1} : \mathbb{R}^d \to \mathbb{R}^{out}$. On appelle une couche non-locale l'opérateur

$$\mathbf{v}_{l+1}(\mathbf{x}) = \mathcal{B}_{\boldsymbol{\theta}}(\mathbf{v})(\mathbf{x}) = \int_D k_\theta(\mathbf{x}, \mathbf{y}, \mathbf{v}_l(\mathbf{x}), \mathbf{v}_l(\mathbf{y}))\mathbf{v}_l(\mathbf{y})d\mathbf{y}$$

ou K_θ le noyau sera la partie apprenable. Dans la majorité des cas l'opérateur est linéaire pour généraliser les couches affines des réseaux classiques.

Definition (Local parametric operator): On se donne une fonction $\mathbf{v}_l : \mathbb{R}^d \to \mathbb{R}^{in}$ et $\mathbf{v}_{l+1} : \mathbb{R}^d \to \mathbb{R}^{out}$. On appelle une couche non-locale l'opérateur

$$\mathbf{v}_{l+1}(\mathbf{x}) = nn_{\boldsymbol{\theta}}(\mathbf{v}_l(\mathbf{x})), \forall \mathbf{x} \in \mathbb{R}^{in}$$

avec $nn_{\boldsymbol{\theta}}$ un réseau de neurones ou une couche paramétrée par $\boldsymbol{\theta}$.

Rappel II

Definition (Opérator paramétrique discret non local): On se donne $\mathbf{v} \in V \subset \left[L^2(\Omega)\right]^p$. On appelle $\mathcal{B}_\theta^h(\mathbf{v})$ un opérateur discret la transformation

$$\mathcal{B}(\mathbf{v})^h(\mathbf{x}) = \frac{1}{n} \sum_{i \in D_h} w_i k_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{y_i}, \mathbf{v}(\mathbf{x}), \mathbf{v}(\mathbf{y_i}))\mathbf{v}(\mathbf{y_i}).$$

Definition (discrétisation d'opérateur convergente): On se donne $\mathbf{v} \in V \subset \left[L^2(\Omega)\right]^p$. On se donne $\mathcal{B}_{\boldsymbol{\theta}}(\mathbf{v})$ et un opérateur discret $\mathcal{B}_\theta^h(\mathbf{v})$. On dit qu'un d'opérateur discret non local est convergent si

$$\| \mathcal{B}_\theta(\mathbf{x}) - \mathcal{B}_\theta^h(\mathbf{x}) \|_{L_\infty(\Omega)} \underbrace{\to}_{n \to \infty} 0, \quad \forall \mathbf{x} \in \Omega, \forall \mathbf{v} \in V$$

Definition (Operateurs neuronals): Un operateur neuronale sera une composition de discrétisation d'opérateurs paramétriques non locaux et locaux ou l'ensemble des donne une discrétisation convergente d'une composition d'opérateur.

2. Operator learning: Attention based methods

Attention I

- On va introduire la dernière architecture de réseau de neurones qui utilise des mécanismes d'attention pour améliorer la modélisation des opérateurs.
- Ici les couche nonlocale ne sont pas linéaires.

Definition (Mécanisme d'attention): Soit $Q \in \mathbb{R}^{q_i, q_o}$ une matrice de requête (query), $K \in \mathbb{R}^{k_i, k_o}$ une matrice de clé (key) et $V \in \mathbb{R}^{v_i, v_o}$ une de valeur (value). On définit le mécanisme d'attention par

$$\mathrm{attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

avec d_k la dimension des clés et softmax la fonction softmax appliquée ligne par ligne. En pratique cela donne:

$$\mathrm{attn}(Q, K, V)_i = \sum_{j=1}^{n} \alpha_{ij} V_j$$

avec

$$\alpha_{ij} = \frac{e^{\frac{\langle Q_i . K_j \rangle}{\sqrt{d_k}}}}{\sum_{l=1}^{n} e^{\frac{\langle Q_i . K_l \rangle}{\sqrt{d_k}}}}$$

Attention II

- Transformation du vecteur **valeur** en fonction de la similarité entre le vecteur de requête et le vecteur des clés.
- Plus la similarité est grande plus la composante vecteur valeur associée à la clé aura d'importance dans le résultat final.

Definition (Cross attention): Soit une matrice d'entrée X et un signal de sortie Y, la cross attention consisteb a appliquer le mécanisme d'attention avec des matrices:

$$\mathrm{Q} = \mathrm{Y}\mathrm{W}_{\mathrm{Q}}, \quad \mathrm{K} = \mathrm{X}\mathrm{W}_{\mathrm{K}}, \quad \mathrm{V} = \mathrm{X}\mathrm{W}_{\mathrm{V}}$$

avec $\mathrm{W}_{\mathrm{Q}}, \mathrm{W}_{\mathrm{K}}, \mathrm{W}_{\mathrm{V}}$ des matrices apprennable.

- Cet opérateur va donc construire des transformations sur le signal de sortie en fonction du signal d'entrée puis construire une matrice de similarité qui va moduler les fonctions valeurs.

Definition (Self attention): Soit une matrice d'entrée X, la self attention consiste a appliquer le mécanisme d'attention avec des matrices:

$$\mathrm{Q} = \mathrm{X}\mathrm{W}_{\mathrm{Q}}, \quad \mathrm{K} = \mathrm{X}\mathrm{W}_{\mathrm{K}}, \quad \mathrm{V} = \mathrm{X}\mathrm{W}_{\mathrm{V}}$$

avec $\mathrm{W}_{\mathrm{Q}}, \mathrm{W}_{\mathrm{K}}, \mathrm{W}_{\mathrm{V}}$ des matrices apprennable.

NO et Attention I

Definition (Continuous nonlocal self-attention based layer): Soit Ω un domaine (qui peut être une variété), $\mathbf{f} : \Omega \to \mathbb{R}^p$, $\mathbf{f} : \Omega \to \mathbb{R}^q$ deux fonctions sur le domaine. On définit une couche nonlocale basée sur l'attention par

$$\mathbf{g}(x) = L_{satt}(\mathbf{f})(x) = \int_\Omega k(x, y, ..)V\mathbf{f}(y)d\mu(y)$$

avec

$$k_{att}(x, y, \mathbf{f}(x), \mathbf{f}(y)) = \frac{e^{\langle W_q \mathbf{f}(x), W_k \mathbf{f}(y)\rangle_{\mathbb{R}^p}}}{\int_\Omega e^{\langle W_q \mathbf{f}(x), W_k \mathbf{f}(z)\rangle_{\mathbb{R}^p}} d\mu(z)}$$

ou

$$k_{att}(x, y, \mathbf{c}(x), \mathbf{f}(y)) = \frac{e^{\langle W_q \mathbf{c}(x), W_k \mathbf{f}(y)\rangle_{\mathbb{R}^p}}}{\int_\Omega e^{\langle W_q \mathbf{c}(x), W_k \mathbf{f}(z)\rangle_{\mathbb{R}^p}} d\mu(z)}$$

avec $W_q \in \mathbb{R}^{p \times d_k}$, $W_k \in \mathbb{R}^{p \times d_k}$ et $V \in \mathbb{R}^{p \times q}$ des matrices apprenables.

- On voit donc que l'attention permet de construire naturellement des opérateurs non locaux à noyaux.
- Premier exemple d'opérateur non local non linéaire.

Multi head attention

Definition (Multi-head attention): Soit h le nombre de têtes d'attention. Soit $\mathbf{f} : \Omega \to \mathbb{R}^p$, $\mathbf{g} : \Omega \to \mathbb{R}^q$ deux fonctions sur le domaine. On définit une couche d'attention multi-tête par

$$\mathbf{g}(x) = \sum_{l=1}^{L} w_l L_{att,l}(\mathbf{f})$$

avec $L_{att,l}(\mathbf{f})$ obtenue par un module d'attention de matrice W_q^l, W_k^l et V^l et $w_l \in \mathbb{R}$ des poids apprenables.

Definition (Nonlocal self-attention based layer): Soit $G = [V, E]$ un graphe avec $|V| = n$. Soit $\mathbf{f} : V \to \mathbb{R}^p$, $\mathbf{g} : V \to \mathbb{R}^q$ deux fonctions sur le graphe. On définit une couche nonlocale basée sur l'attention par

$$\mathbf{g}(i) = \sum_{j=1}^{n} w_i k_{att}(i,j) V \mathbf{f}(j)$$

avec

$$k_{att}(i,j,\mathbf{f}(i),\mathbf{f}(j)) = \frac{e^{\langle W_q \mathbf{f}(i), W_k \mathbf{f}(j)\rangle_{\mathbb{R}^p}}}{\sum_{l=1}^{n} e^{\langle W_q \mathbf{f}(i), W_k \mathbf{f}(l)\rangle_{\mathbb{R}^p}}}, \quad \text{ou} \quad k_{att}(i,j,\mathbf{c}(i),\mathbf{f}(j)) = \frac{e^{\langle W_q \mathbf{c}(i), W_k \mathbf{f}(j)\rangle_{\mathbb{R}^p}}}{\sum_{l=1}^{n} e^{\langle W_q \mathbf{c}(i), W_k \mathbf{f}(l)\rangle_{\mathbb{R}^p}}}$$

avec $W_q \in \mathbb{R}^{p \times d_k}$, $W_k \in \mathbb{R}^{p \times d_k}$ et $V \in \mathbb{R}^{p \times q}$ des matrices apprenables.

Neural operator et couche locale

Proposition (Attention nonlocal neural operator layer): Soit Ω un domaine (qui peut être une variété), $\mathbf{f} : \Omega \to \mathbb{R}^p$, $\mathbf{g} : \Omega \to \mathbb{R}^q$ deux fonctions sur le domaine. Soit des ensemble de points de discretisation $X = \{\mathbf{x}_1, ..., \mathbf{x}_n\} \subset \Omega$ et $Y = \{\mathbf{y}_1, ..., \mathbf{y}_m\} \subset \Omega$ obtenue à partir d'une lois de probabilité $p(y)$. Les couches de "self" et "cross" définissent des approximations convergentes des couches continues lorsque le nombre de points de discretisation tend vers l'infini a condition que

$$w_i = \frac{1}{p(\mathbf{y}_i)}$$

- Pour obtenir un opérateur neuronal complet on peut combiner les couches d'attention avec des couches locales non linéaires.
- Entre les couches d'attention les transformers utilisent des couches feedforward classiques combinées à des couches de normalisation.
- On peut imaginer des architectures similaires pour les opérateurs neuronaux basés sur l'attention.

Couche locale

Definition (Local normalization layer): Soit Ω un domaine (qui peut être une variété), $\mathbf{f} : \Omega \to \mathbb{R}^p$ une fonction sur le domaine. On définit une couche de normalisation L_{ln} locale est donnée par

$$\mathbf{g}_k(x) = \gamma_k \frac{\mathbf{f}_k(x) - \mu(\mathbf{f}(x))}{\sqrt{\sigma^2(\mathbf{f}(x)) + \varepsilon}} + \beta_k, \forall k \in \{1, ..., p\}$$

avec

$$\mu(\mathbf{f}(x)) = \frac{1}{p}\sum_{i=1}^{p} \mathbf{f}_i(x), \quad \sigma^2(\mathbf{f}(x)) = \frac{1}{p}\sum_{i=1}^{p} (\mathbf{f}_i(x) - \mu(\mathbf{f}(x)))^2$$

et $\boldsymbol{\gamma} = (\gamma_1, ..., \gamma_p)$, $\boldsymbol{\beta} = (\beta_1, ..., \beta_p)$ des paramètres apprennable et ε un petit scalaire pour la stabilité numérique.

Neural operator block

Definition (Transformer neural operator block): Soit Ω un domaine (qui peut être une variété), $\mathbf{f} : \Omega \to \mathbb{R}^p$, $\mathbf{g} : \Omega \to \mathbb{R}^q$ deux fonctions sur le domaine. On définit bloc de transformer neural operator par la succession des opérations suivantes:

$$\mathbf{g}_1(x) = L_{att}(\mathbf{f}) + W_1\mathbf{f}(x)$$

$$\mathbf{g}_2(x) = L_{ln}(\mathbf{g}_1)(x)$$

$$\mathbf{g}_3(x) = W_2\mathbf{g}_2(x) + L_{nn}(\mathbf{g}_2(\mathbf{x}))$$

et

$$\mathbf{g}(x) = L_{ln}(\mathbf{g}_3)(x)$$

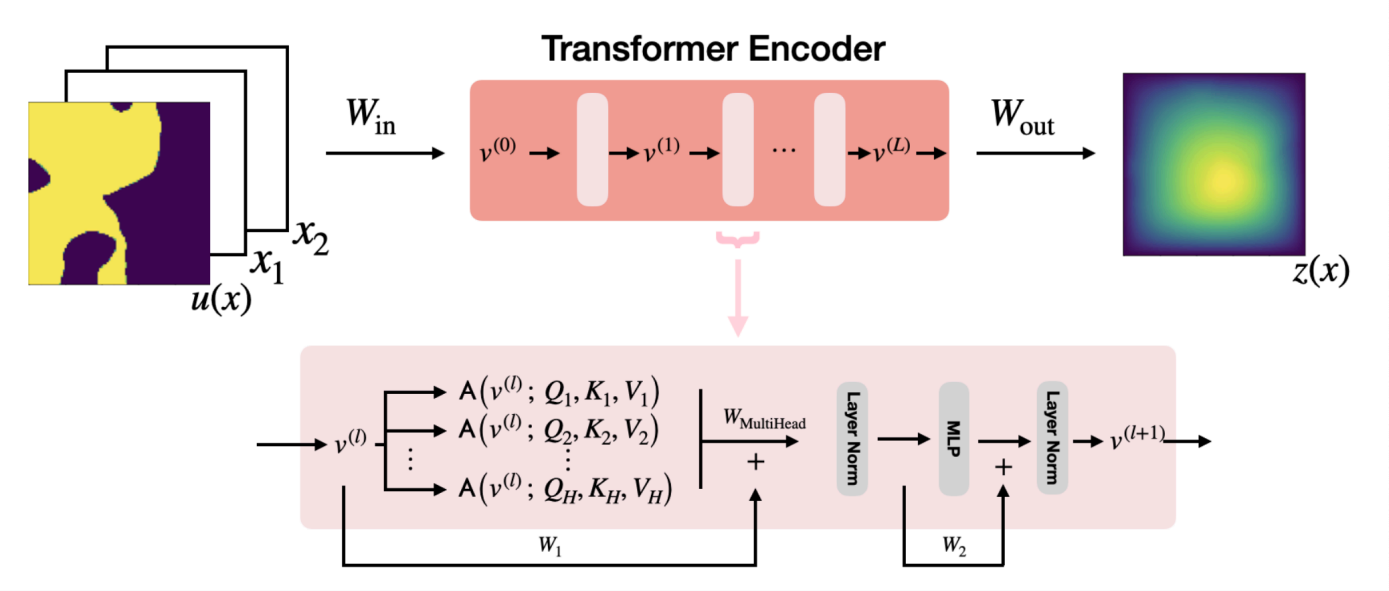
avec W_1, W_2 des matrices apprenables, L_{att} une couche d'attention (self ou cross), L_{ln} une couche de normalisation locale et L_{nn} une couche feedforward non linéaire.

Neural operator block

Definition (Transformer neural operator): Soit Ω un domaine (qui peut être une variété), $\mathbf{f} : \Omega \to \mathbb{R}^p$, $\mathbf{g} : \Omega \to \mathbb{R}^q$ deux fonctions sur le domaine. On définit un "transformer neural operator" par la transformation:

$$\mathbf{g}(x) = Q \circ L_{1,Bt} \circ ....L(L, Bt) \circ P(\mathbf{f})(x)$$

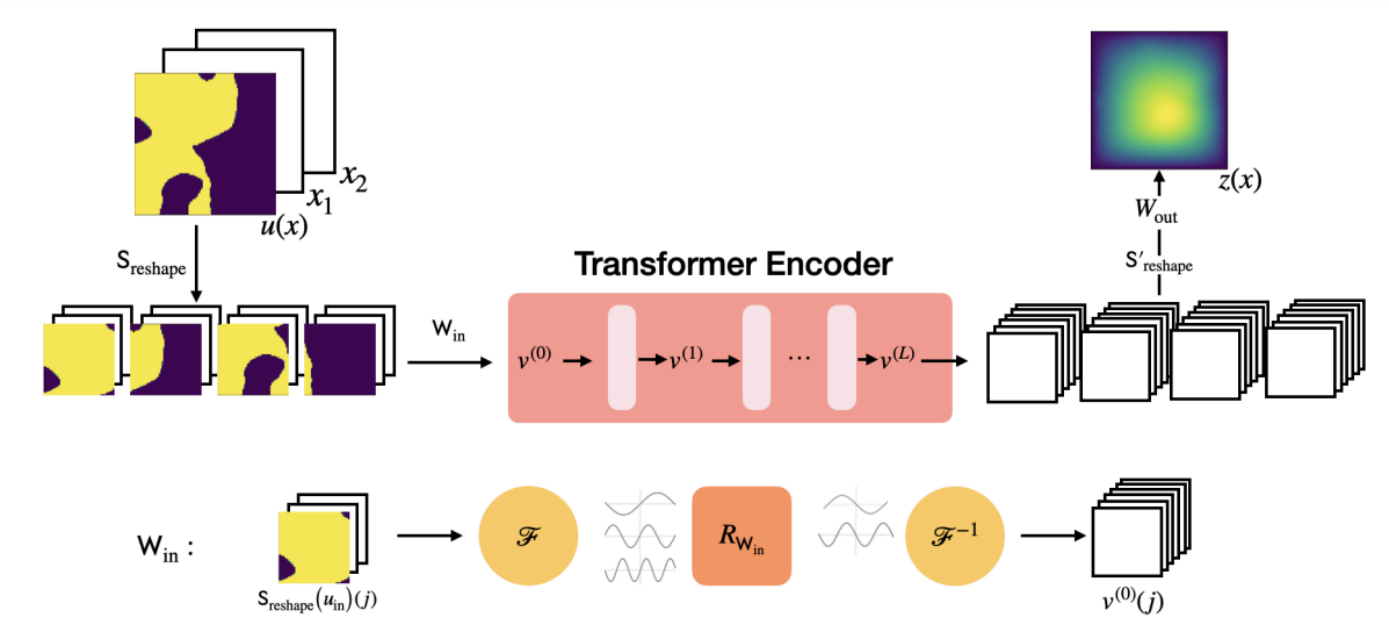
avec $L(i, Bt)$ des blocs de type transformer, P et Q des couches linéaires locales d'adaptation de dimension.



Patch Transformer neural operator

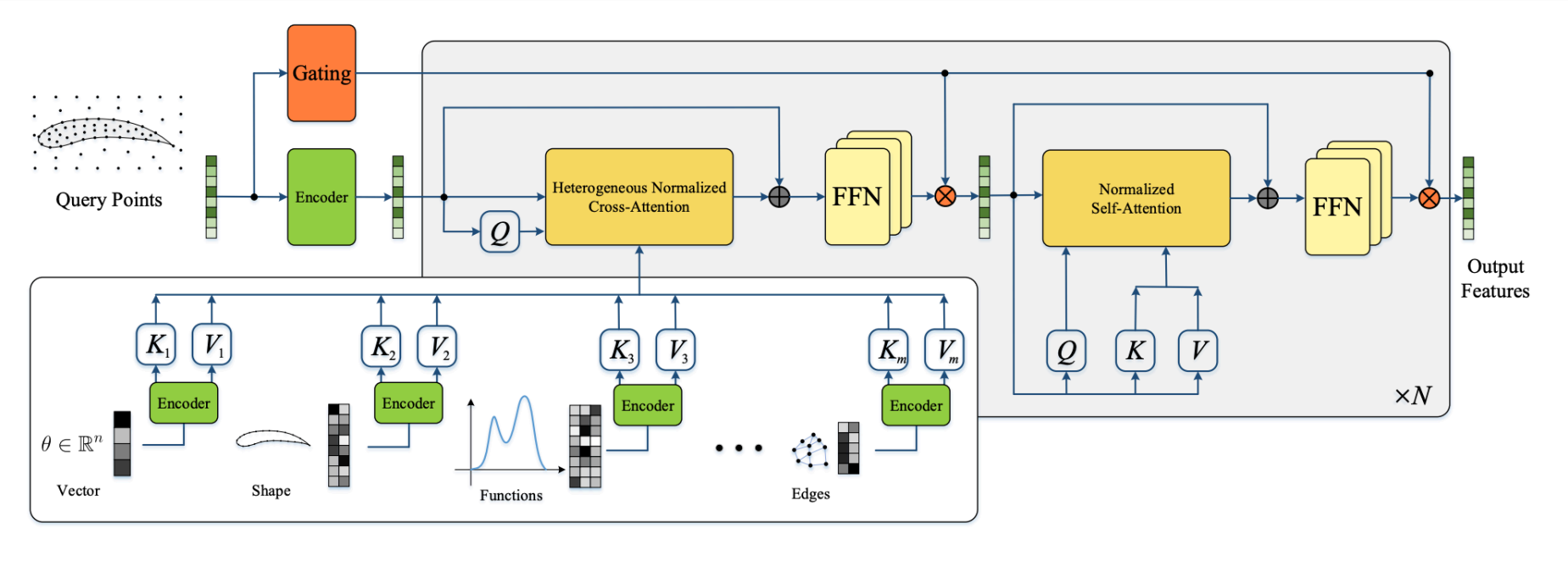
Idea: Pour réduire le coût computationnel on peut imaginer découper le domaine en patchs/sous domaines et appliquer un transformer sur les représentations locales de chaque patch.

- Ils peuvent être recombinés entre eux par les couches de multi-head attention.



GNOT I

- On va maintenant introduire une autre architecture basée sur l'attention: le GNOT (General Neural Operator Transformer).



- On cherche un opérateur:

$$\mathcal{G} : \mathcal{A} \to \mathcal{H}$$

avec $\mathcal{A} = \mathcal{H} \times ... \times \mathcal{H} \times \mathbb{R}^p$ qui va continue les **points du domaine**, les fonctions d'entrées et de sortie du modèle (source, condition aux limites, etc) et les paramètres physiques.

GNOT II

- L'objectif est de pouvoir traiter des entrées variées: les points du domaine, les conditions aux limites, les sources, les paramètres physiques.
  - Soit $\beta \in \mathbb{R}^p$. On le transforme avec un encodeur MLP $f_{\theta_1}(\beta) \in \mathbb{R}^{1,n_e}$
  - Soit $\mathbf{x}_b \in \mathbb{R}^{n_x,d}$ les points de bord du domaine. On les transforme avec un encodeur MLP $f_{\theta_2}(\mathbf{x}) \in \mathbb{R}^{n_x d}$
  - Soit $\mathbf{x}, \mathbf{a} \in \mathbb{R}^{n_x,d+p}$ les points du domaine et les valeurs des fonctions d'entrée à ses points. On les transforme avec un encodeur MLP $f_{\theta_3}(\mathbf{x}, \mathbf{a}) \in \mathbb{R}^{n_d d}$

Remark: On peut encoder d'autres quantités comme des quantités aux arrêtes etc.

- On va encoder $(\mathbf{x}, \mathbf{a})$ avec un encodeur MLP $f_{\theta_e}(\mathbf{x}, \mathbf{a}) \in \mathbb{R}^{n,n_e}$. On le nomme X
- L 'encodage des quantit"s supplémentaires, comme les paramètres, les points de bords ou autre est nommé $Y \in \mathbb{R}^{n,n_e}$

Idea: On va utiliser une architecture basée sur l'attention cross entre X et Y pour construire un opérateur non local entre les deux.

GNOT III

Idea: Après l'étape d'encodage on va appliquer des opérateurs d'attention.

- Cross-attention sur X et Y:

$$\mathbf{z}_i = \sum_{j=1}^{n_d} \frac{e^{\mathbf{q}_i \cdot \mathbf{k}_j}}{\sum_{l=1}^{n} e^{\mathbf{q}_i \cdot \mathbf{k}_l}} \mathbf{v}_j$$

avec $\mathbf{k}_j = W_k \mathbf{y}_j$ et $\mathbf{v}_j = W_v \mathbf{y}_j$

Il est proposé une simplification:

- ▸ On normalise:

$$\tilde{\mathbf{q}}_i = \frac{e^{q_{ij}}}{\sum_{l=1}^{n_e} e^{q_{il}}}, \qquad \tilde{\mathbf{k}}_j = \frac{e^{k_{ji}}}{\sum_{l=1}^{n_e} e^{k_{li}}}$$

- ▸ La nouvelle attention est donnée attention est donnée par:

$$\mathbf{z}_i = \sum_{j=1}^{n_d} \frac{\tilde{\mathbf{q}}_i \cdot \tilde{\mathbf{k}}_j}{\sum_{l=1}^{n_e} \tilde{\mathbf{q}}_i \cdot \tilde{\mathbf{k}}_l} \mathbf{v}_j = \alpha_i \sum_{j=1}^{n_d} \tilde{\mathbf{q}}_i \cdot \tilde{\mathbf{k}}_j \mathbf{v}_j, \quad \text{avec} \quad \alpha_i = \frac{1}{\sum_{l=1}^{n_e} \tilde{\mathbf{q}}_i \cdot \tilde{\mathbf{k}}_l} = \alpha_i \tilde{\mathbf{q}}_i \cdot \sum_{j=1}^{n_d} \tilde{\mathbf{k}}_j \otimes \mathbf{v}_j$$

GNOT III

Definition (cross-attention layer of GNOT): On définit notre signal d'entrée après encodage $X \in \mathbb{R}^{N,n_e}$ et les signaux supplémentaires $Y_{1\leq l\leq L} \in \mathbb{R}^{N,n_e}$. La couche d'attention cross de GNOT est donnée par

$$z_i = \tilde{q}_i + \frac{1}{L}\sum_{l=1}^{L} \alpha_{i,l}\tilde{q}_{i,l} \cdot \left(\sum_{j=1}^{N} \tilde{k}_j \otimes v_j\right)$$

- Pour traiter les aspects multi-échelles les auteurs proposent des construire de pondérations depdante de l'espace de notre état latent modifié par plusieurs MLP.
- On a donc

$$z_i = z_i + \sum_{k=1}^{K} p(x_i)\mathrm{MLP}_i(z_i)$$

avec

$$p(x_i) = \frac{\exp^{G_k(x_i)}}{\sum_{l=1}^{K}\exp^{G_l(x_i)}}, \qquad G : \mathbb{R}^d \to \mathbb{R}^K$$

3. Operator learning: Time problem extension

Apprentissage d'opérateurs temporel

- Soit un problème elliptique de la forme

$$\begin{cases} \partial_t u - \Delta u = f(t, \mathbf{x}), & \mathbf{x} \in \Omega \\ u(t, \mathbf{x}) = 0 & \mathbf{x} \in \partial\Omega \\ u(0, \mathbf{x}) = u_{\{0\}}(\mathbf{x}) & \mathbf{x} \in \Omega \end{cases}$$

- On voudrait prédire rapidement une solution pour un $f(t, \mathbf{x})$ ou un $u_0(\mathbf{x})$ généraux.
- Si la solution existe, on peut formellement dire qu'il existe $G : \mathcal{H}_x \to \mathcal{H}_y$ tel que

$$u(.) = G^{-1}(f(.), u_0(.))$$

- Comme précédemment peut généraliser en cherchant à approcher $G^+(\mathbf{a}(.)) = u(.)$ avec $\mathbf{a}$ l'ensemble des fonctions qui sont des entrées du problème (sources, conditions limites, conditions initiales, coefficients de l'EDP).
- Si on regarde précisément le problème:

$$u(t, .) = G^{-1}(u_0(.))$$

- On voit qu'il s'agit en faite d'apprendre le flot d' EDP.

Remark (Deux cas): flot discret ou flot continu en temps

Opérateurs neuronaux continus en temps

- Peut t'on généraliser la théorie de Green ? Oui

$$u(t,\mathbf{x}) = \int_\Omega G(t,0,\mathbf{x},\mathbf{y})u_0(\mathbf{y})d\mathbf{y} + \int_0^t \int_\Omega G(t,\tau,\mathbf{x},\mathbf{y})f(\tau,\mathbf{y})d\mathbf{y}d\tau$$

$$+ \int_0^t \int_{\partial\Omega} [u(t,\mathbf{x})\nabla G(t,0,\mathbf{x},\mathbf{y}) - G(t,0,\mathbf{x},\mathbf{y})\nabla u(t,\mathbf{x})]d\mathbf{y}d\tau$$

avec

$$(\partial_t - \Delta)G(t,\tau,\mathbf{x},\mathbf{y}) = \delta(\mathbf{x}-\mathbf{y})\delta(t-\tau)$$

et des conditions initiales/bords nulles.

Definition (GreenNet temporal operator): Il s'agit d'un réseau à une un couche sans couche d'extrapolation et de projection ou la couche principale est $G_\theta : H^{s(\Omega)^p} \to H^{s(\Omega)^p}$ est paramétrée de la façon suivante:

$$G_{\boldsymbol{\theta}}(\mathbf{f},\mathbf{u}_0) = \int_\Omega k_{\boldsymbol{\theta}}(t,0,\mathbf{x},\mathbf{y})\mathbf{u}_0(\mathbf{y})d\mathbf{y} + \int_0^t \int_\Omega k_{\boldsymbol{\theta}}(t,\tau,\mathbf{x},\mathbf{y})f(\tau,\mathbf{y})d\mathbf{y}d\tau$$

avec $k_{\boldsymbol{\theta}}$ un réseau de type MLP, Fourier etc.

- Les discretisations convergentes sont similaires au cas stationnaire.

Low rank approximation et deepOnet

Definition (GreenNet temporel de Bas rang): Soit $\mathbf{f}$ une fonction uniquement spatiale (condition initiale) ou dépendent du temps (source). On propose de prendre:

$$G_{\boldsymbol{\theta}}(\mathbf{f}) = \sum_{k=1}^{R} \langle f, \psi_i \rangle_{L^2} \varphi_{\boldsymbol{\theta},k}(\mathbf{x}, t)$$

avec

$$\langle f, \psi_i \rangle_{L^2} = \frac{1}{m} \sum_{j=1}^{m} \psi_{\boldsymbol{\theta},i}(\mathbf{x}_j, t_j) f(\mathbf{x}_j, t_j), \quad \text{ou} \quad \langle f, \psi_i \rangle_{L^2} = \frac{1}{m} \sum_{j=1}^{m} \psi_{\boldsymbol{\theta},i}(\mathbf{x}_j, t_j) f(\mathbf{x}_j),$$

Definition (DeepOnet temporel): Soit $\mathbf{a}$ une fonction uniquement spatiale (condition initiale) ou dépendent du temps (source). On propose de prendre:

$$G_{\boldsymbol{\theta}}(\mathbf{a}(t_1, \mathbf{x}_1), ...\mathbf{a}(t_m, \mathbf{x}_m)) = \sum_{i=1}^{R} \alpha_i b_{\boldsymbol{\theta},i}(\mathbf{x}, t)$$

avec $b_{\boldsymbol{\theta}}$ un réseau de type MLP de $\mathbb{R}^d$ dans $\mathbb{R}^R$ et

$$t_{\boldsymbol{\theta}}(\mathbf{a}(t_1, \mathbf{x}_1), ...\mathbf{a}(t_m, \mathbf{x}_m)) \to (\alpha_1, ..., \alpha_K)$$

un réseau de $\mathbb{R}^{q \times m}$ dans $\mathbb{R}^R$

FNO continue en temps

- Une couche de Fourier est donnée par:

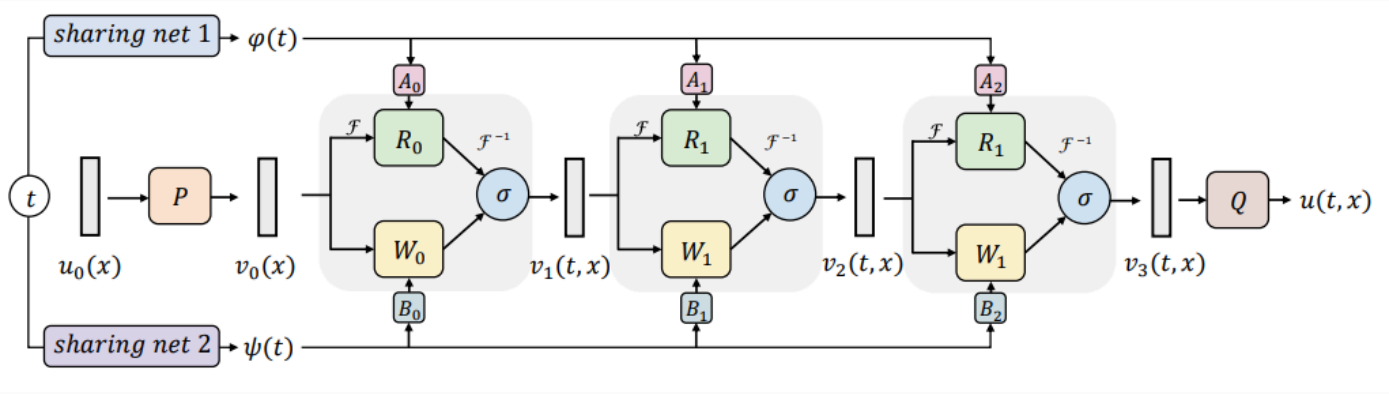
$$\mathbf{v}_{l+1} = \sigma\Big(\mathrm{W}_l\mathbf{v}_l + \Big(\mathcal{F}^{\{-1\}} \circ \hat{\kappa}_{\theta_l} \circ \mathcal{F}\Big)(\mathbf{v}_l)\Big)$$

- Pour l'étendre au temporel il est proposé une modulation temporelle des poids de chaque couche:

$$\mathbf{v}_{l+1} = \sigma\Big(\psi_{\theta_t}(t)\mathrm{W}\mathbf{v}_l + \Big(\mathcal{F}^{-1} \circ \varphi_{\theta_t}(t,\mathbf{k})\hat{\kappa}_{\theta_l} \circ \mathcal{F}\Big)(\mathbf{v}_l)\Big)$$

avec $\mathbf{k}$ les fréquences.

- Il est proposé de prendre $\varphi_{\boldsymbol{\theta}_t}(t,\mathbf{k}) = \varphi(t)\mathrm{A}_{l(\mathbf{k})}$ avec $\mathrm{A}(\mathbf{k})$ une matrice entrainable.
- Il est proposé de prendre $\varphi_{\boldsymbol{\theta}_t}(t,\mathbf{k}) = \mathrm{diag}(\psi(t)\mathrm{B}_l)$ avec B_l une matrice entrainable.
- les fonctions $\psi(t)$ et $\varphi(t)$ sont communes à toutes les couches.



Transformeur continue en temps

Idea (Transformer) : pour prendre un transformer temporel la méthode la plus utilisé consiste a moduler les couches locales par des fonctions temporelles

- On peut imaginer faire dependre les MLP locales du temps.
- On peut moduler les couches linéaires:

$$\mathbf{g}(x, t) = \psi_{\theta_t}(t)W$$

- On peut moduler les couches de normalisation locale:

$$\left(F_{\text{LayerNorm}}(\mathbf{v}; \gamma, \beta,)(\mathbf{x})\right)_k = \gamma_{k(t)} \cdot \frac{(\mathbf{v}(\mathbf{x}))_k - m(\mathbf{v}(\mathbf{x}))}{\sqrt{\sigma^2(\mathbf{v}(\mathbf{x})) + \varepsilon}} \beta_{k(t)},$$

avec

- $\beta(t) = \beta_1 + \beta_2 t$ et $\gamma(t) = \gamma_1 + \gamma_2 t$
- $\beta(t), \gamma(t)$ des réseaux de neurones type MLP.

Remark : On pourrait moduler les matrices d'attention par une fonction $\psi_{\theta_t}(t)$.

Propriété de semi-groupe

Remark (Propriété de semi-groupe): Si on cherche a prédire la solution a partir de la condition initiale sans source. Le flot qui sera approchée par l'opérateur neuronale satisfait la **structure de semi-groupe**.

- On se sait pas imposer cette structure dans les architectures actuelles. On peut imposer cettes structure faiblement.
  - La fonction de coût classique est donnée par:

$$\mathcal{L}(\boldsymbol{\theta}) := \frac{1}{M(K+1)} \sum_{i=1}^{M} \sum_{k=0}^{K} \| \, G(t_k, a_i(\mathbf{x})) - u_i(t_k, \mathbf{x}) \|_{L^p(D)}^p,$$

avec $u_i(t_k, \mathbf{x})$ la solution associée à la condition initiale $a_i(\mathbf{x})$;

- On impose faiblement la structure de semi-groupe:

$$\hat{\mathcal{L}}(\boldsymbol{\theta}) := \frac{1}{M\hat{K}} \sum_{i=1}^{M} \sum_{k,\hat{k}=0, k\leq\hat{k}}^{K} \| \, u_i(t_{\hat{k}}, \mathbf{x}) - G_{\boldsymbol{\theta}}(t_{\hat{k}} - t_k, u_i(t_k, \mathbf{x})) \, \|_{L^p(D)}^p,$$

Remark: Une autre approche est de considérer une architecture de type EDO

$$\frac{d}{dt}\mathbf{u}(t) = \mathcal{N}_{\boldsymbol{\theta}}(\mathbf{u}(t))$$

avec $\mathcal{N}_{\boldsymbol{\theta}}(\mathbf{u}(t))$ un opérateur neuronale.

Operateur neuronaux discret en temps I

Idea : On demande à l'opérateur neuronale de prédire la solution à des temps discret.

- On se donne un opérateur neuronale $G_{(\theta(u)}$
- Plusieurs options:
  - On prédit un temps avec un temps: $u_{t_{k+1}}(x) = G_\theta\left(u_{t_k}(x)\right)$
  - On prédit plusieurs temps avec un temps: $\left(u_{t_{k+m}}(x), ..., u_{t_{k+1}}(x)\right) = G_\theta\left(u_{t_k}(x), ..., u_{t_{k-m}}(x)\right)$

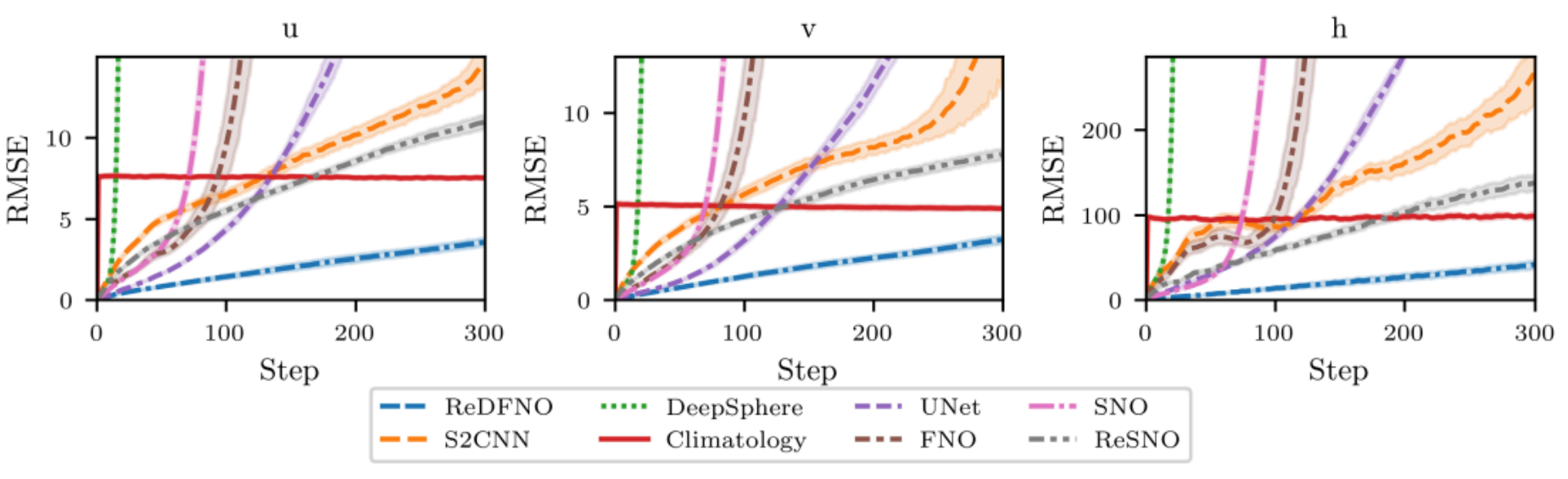
Definition (NO auto-régressif) : On utilise donc les opérateurs neuronaux de façon auto-régressif:

$$u_{t_n}(x) = (G_\theta \circ ... \circ G_\theta)(u_0(x))$$

Remark (Inconvénient) : Est ce ce type de prédiction est **stable** en temps ?

Operateur neuronaux discret en temps II

- Problème de stabilité en temps long.

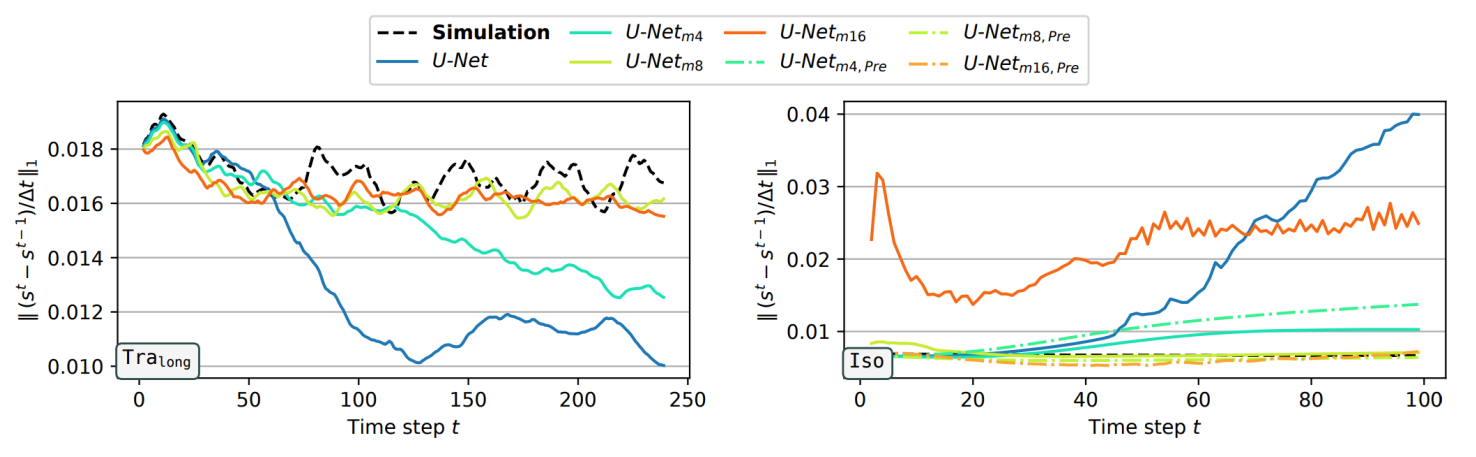


Idea : Un entrainement sur plusieurs pas de temps (unrolled) peut stabiliser les opérateurs neuronaux.

- Références:
  - “Benchmarking Autoregressive Conditional Diffusion Models for Turbulent Flow Simulation”. N. Thuerey and al.
  - “Towards Stability of Autoregressive Neural Operators”. J. Brown and al.

Operateur neuronaux discret en temps II

- Comparaison de different unrolling pour un Unet



- Coût mémoire et CPU de “l’unrolling”