Kernel integral operator by wavelet

Ref: Multiwavelet-based Operator Learning for Differential Equations

Suwei Yang 2023.09.01

Outline

- Problem set
- Wavelet transformation v.s. Fourier transformation
- Math procedure
- Architecture
- Evaluation
- Conclusion

Problem set

$$Ta(x) = \int_D K(x,y)a(y)dy.$$

Solve it, solve map problem

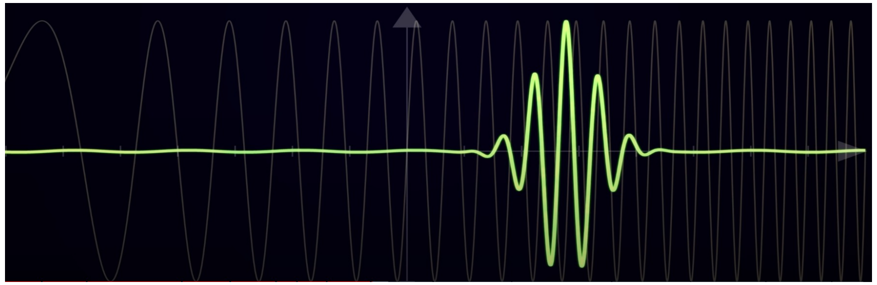
Transform	Representation	Input
Fourier transform	$\hat{X}(f) = \int_{-\infty}^{\infty} x(t)e^{-i2\pi ft}\,dt$	f : frequency
Time–frequency analysis	$X(t, f)$	t time; f frequency
Wavelet transform	$X(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{\infty} \overline{\Psi\left(\frac{t-b}{a}\right)}\,x(t)\,dt$	a scaling ; b time shift factor

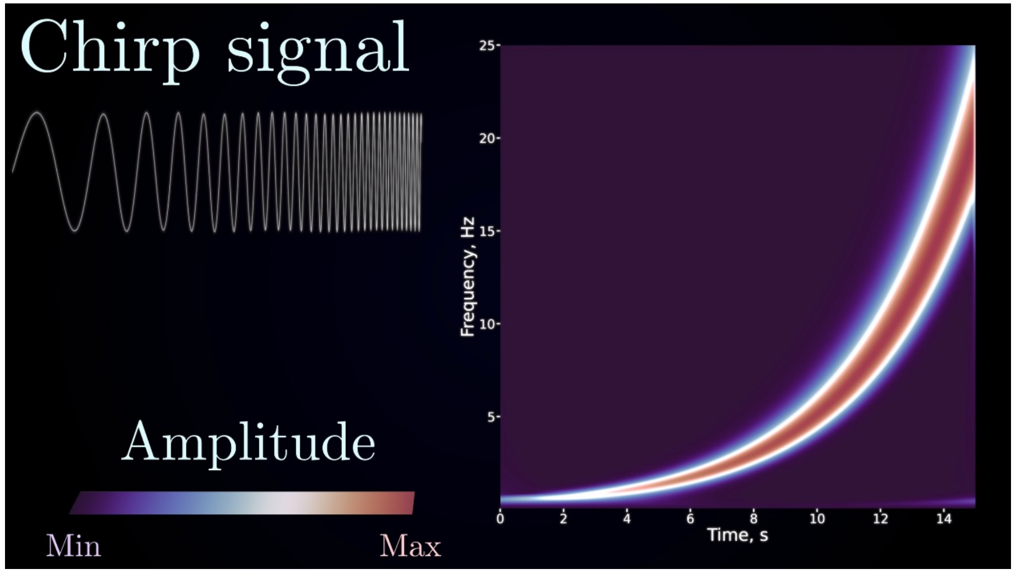
Wavelet transformation v.s. Fourier transformation

- Fourier: 1D (time) -> 1D (frequency)
- Wavelet: 1D(time) -> 2D (frequency + time)
- Tradeoff: time , frequency

Wavelet
t: time
b: shift (time)
a: scale (frequency)

$$\Psi_{a,b} = \Psi\left(\frac{t-b}{a}\right)$$





Math procedure

- Evaluate Ta=u

$$Ta(x) = \int_D K(x,y)a(y)dy.$$

$$T_n = \sum_{i=L+1}^{n} (Q_i T Q_i + Q_i T P_{i-1} + P_{i-1} T Q_i) + P_L T P_L.$$

$$A_i = Q_i T Q_i \qquad B_i = Q_i T P_{i-1} \qquad C_i = P_{i-1} T Q_i \qquad \bar{T} = P_L T P_L$$

$$\mathrm{s}, \mathrm{d} = mwt(a(x))$$

$$U_{dl}^n = A_n d_l^n + B_n s_l^n, \qquad U_{\hat{s}l}^n = C_n d_l^n, \qquad U_{sl}^L = \bar{T} s_l^L$$

$$u = inverse\_mwt(U_{dl}^n, U_{\hat{s}l}^n + U_{sl}^L)$$

$$\mathbf{s}_l^n = H^{(0)} \mathbf{s}_{2l}^{n+1} + H^{(1)} \mathbf{s}_{2l+1}^{n+1},$$
$$\mathbf{d}_l^n = G^{(0)} \mathbf{s}_{2l}^{n+1} + G^{(1)} \mathbf{s}_{2l+1}^{n+1}.$$

Decompose

$$\mathbf{s}_{2l}^{n+1} = \Sigma^{(0)} (H^{(0)\,T} \mathbf{s}_l^n + G^{(0)\,T} \mathbf{d}_l^n),$$
$$\mathbf{s}_{2l+1}^{n+1} = \Sigma^{(1)} (H^{(1)\,T} \mathbf{s}_l^n + G^{(1)\,T} \mathbf{d}_l^n).$$

Reconstruct

Math procedure - material

Decompose

$$\mathbf{s}_l^n = H^{(0)}\mathbf{s}_{2l}^{n+1} + H^{(1)}\mathbf{s}_{2l+1}^{n+1},$$
$$\mathbf{d}_l^n = G^{(0)}\mathbf{s}_{2l}^{n+1} + G^{(1)}\mathbf{s}_{2l+1}^{n+1}.$$

Reconstruct

$$\mathbf{s}_{2l}^{n+1} = \Sigma^{(0)}(H^{(0)T}\mathbf{s}_l^n + G^{(0)T}\mathbf{d}_l^n),$$
$$\mathbf{s}_{2l+1}^{n+1} = \Sigma^{(1)}(H^{(1)T}\mathbf{s}_l^n + G^{(1)T}\mathbf{d}_l^n).$$

$$\mathbf{s}_l^n = [\langle f, \phi_{il}^n\rangle_{\mu_n}]_{i=0}^{k-1} \quad \mathbf{d}_l^n = [\langle f, \psi_{il}^n\rangle_{\mu_n}]_{i=0}^{k-1}$$

$$\phi_{jl}^n(x) = 2^{n/2}\phi_j(2^n x - l), \quad j = 0, 1, \dots, k-1, \quad l = 0, 1, \dots, 2^n - 1, \text{w.r.t.} \quad \mu_n$$

$$\phi_i = \sqrt{2i+1}P_i(2x-1)$$

$$H_{ij}^{(0)} = \sqrt{2}\int_0^{1/2} \phi_i(x)\phi_j(2x)w(2x)dx, \quad G_{ij}^{(0)} = \sqrt{2}\int_0^{1/2} \psi_i(x)\phi_j(2x)w(2x)dx, \quad \Sigma_{ij}^{(0)} = 2\int_0^{1/2} \phi_i(2x)\phi_j(2x)w(x)dx,$$
$$H_{ij}^{(1)} = \sqrt{2}\int_{1/2}^{1} \phi_i(x)\phi_j(2x-1)w(2x-1)dx \quad G_{ij}^{(1)} = \sqrt{2}\int_{1/2}^{1} \psi_i(x)\phi_j(2x-1)w(2x-1)dx \quad \Sigma_{ij}^{(1)} = 2\int_{1/2}^{1} \phi_i(2x-1)\phi_j(2x-1)w(x)dx$$

Filters

$$\psi_i \leftarrow \phi_i^{(1)} - \textstyle\sum_{j=0}^{m-1}\langle\phi_i^{(1)}, \phi_j^{(0)}\rangle_{\mu_0}\phi_j^{(0)} - \sum_{l=0}^{i-1}\langle\phi_i^{(i)}, \psi_l\rangle_{\mu_0}\psi_l,$$
$$\psi_i \leftarrow \frac{\psi_i}{||\psi_i||_{\mu_0}}.$$

Gram-Schmidt Orthogonalization (GSO)

$$\sum_{i=1}^{n}\omega_i f(x_i) = \int_a^b f(x)w(x)dx$$

$$\omega_i = \frac{a_n}{a_{n-1}}\frac{\int_a^b P_{n-1}^2(x)w(x)dx}{P_n'(x_i)P_{n-1}(x_i)}$$

$$iP_i(x) = (2i-1)xP_{i-1}(x) - (i-1)P_{i-2}(x)$$
$$(2i+1)P_i(x) = P_{i+1}'(x) - P_{i-1}'(x),$$

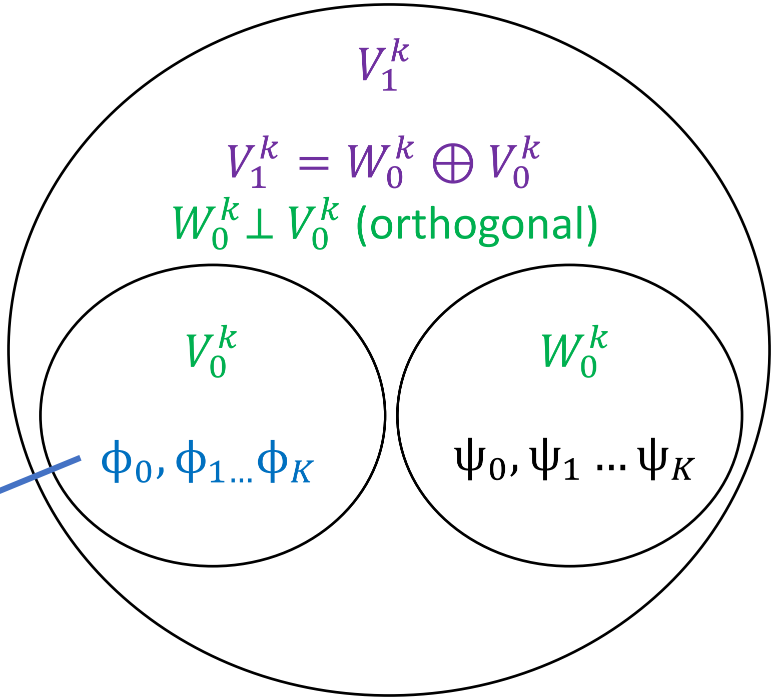
Gaussian Quadrature

Math

$$V_0^k \subset V_1^k \subset \cdots \subset V_{n-1}^k \subset V_n^k$$

Example:
Legendre polynomials

n	$P_n(x)$
0	1
1	x
2	$\frac{1}{2}\left(3x^2-1\right)$
3	$\frac{1}{2}\left(5x^3-3x\right)$
4	$\frac{1}{8}\left(35x^4-30x^2+3\right)$
5	$\frac{1}{8}\left(63x^5-70x^3+15x\right)$
6	$\frac{1}{16}\left(231x^6-315x^4+105x^2-5\right)$
7	$\frac{1}{16}\left(429x^7-693x^5+315x^3-35x\right)$
8	$\frac{1}{128}\left(6435x^8-12012x^6+6930x^4-1260x^2+35\right)$
9	$\frac{1}{128}\left(12155x^9-25740x^7+18018x^5-4620x^3+315x\right)$
10	$\frac{1}{256}\left(46189x^{10}-109395x^8+90090x^6-30030x^4+3465x^2-63\right)$

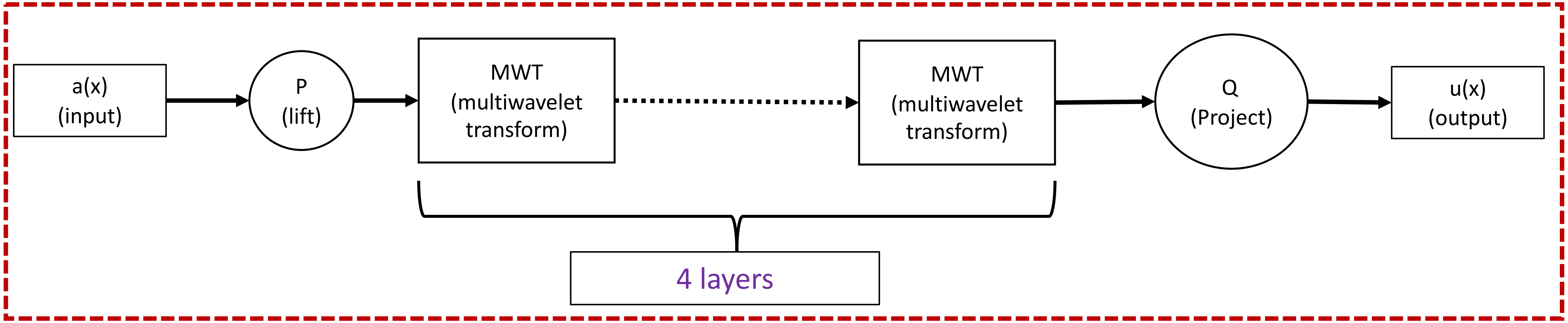


$$\omega_i = \frac{a_n}{a_{n-1}} \frac{\int_a^b P_{n-1}^2(x) w(x) dx}{P_n'(x_i) P_{n-1}(x_i)}$$

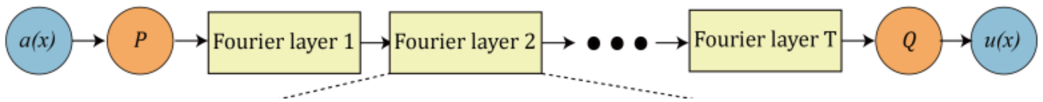
$$\psi_i \leftarrow \phi_i^{(1)} - \sum_{j=0}^{m-1} \langle \phi_i^{(1)}, \phi_j^{(0)} \rangle_{\mu_0} \phi_j^{(0)} - \sum_{l=0}^{i-1} \langle \phi_i^{(i)}, \psi_l \rangle_{\mu_0} \psi_l,$$

$$\psi_i \leftarrow \frac{\psi_i}{||\psi_i||_{\mu_0}}.$$

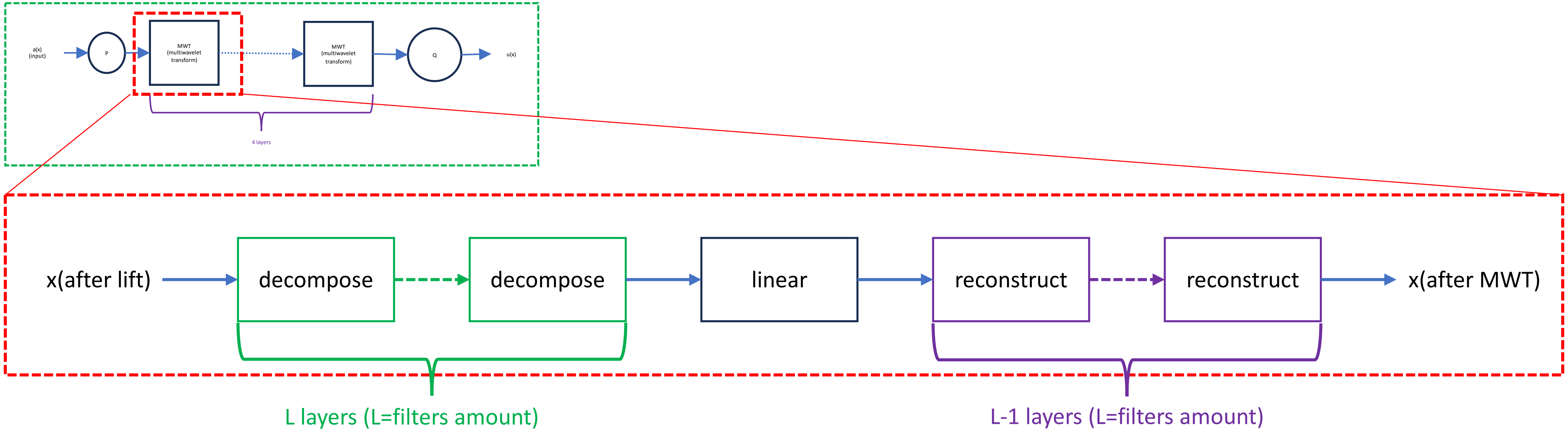
Architecture



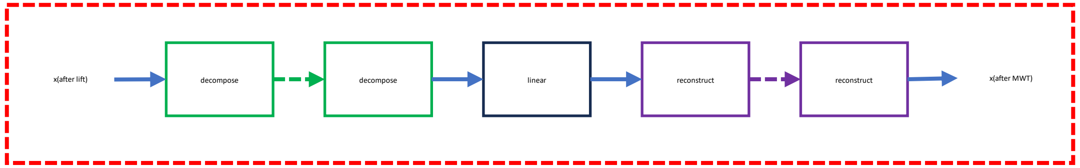
Compare with FNO
(T=4)



Architecture-MWT



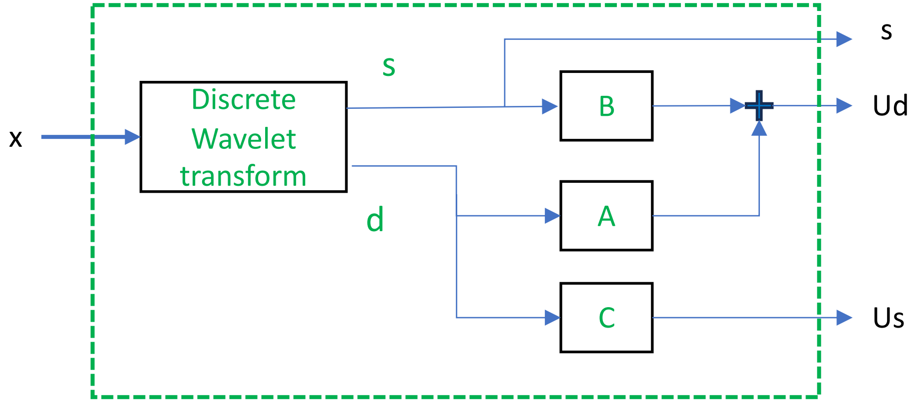
Architecture-decompose & reconstruct

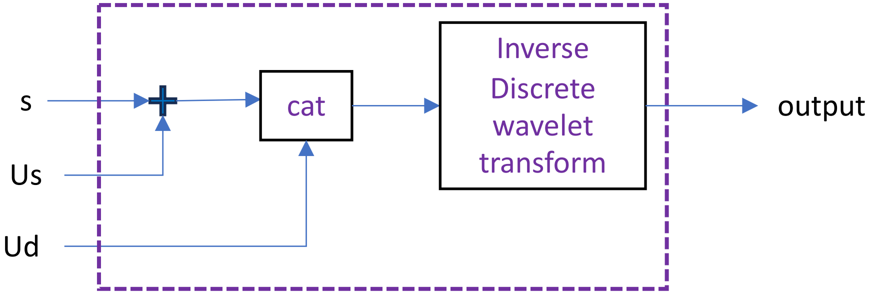



```python
#         decompose
        for i in range(ns-self.L):
            d, x = self.wavelet_transform(x)
            Ud += [self.A(d) + self.B(x)]
            Us += [self.C(d)]
        x = self.T0(x.view(B, 2**self.L, 2**self.L, -1)).view(
            B, 2**self.L, 2**self.L, c, ich) # coarsest scale transform

#        reconstruct
        for i in range(ns-1-self.L,-1,-1):
            x = x + Us[i]
            x = torch.cat((x, Ud[i]), -1)
            x = self.evenOdd(x)

        return x
```

Decompose
A: NN
B: NN
C: NN
s: result from high frequency filters
d: result from low frequency filters
Ud: coefficient from multiwavelet
Us: coefficient from multiwavelet

Reconstruct
Cat: concatenates given data





Result

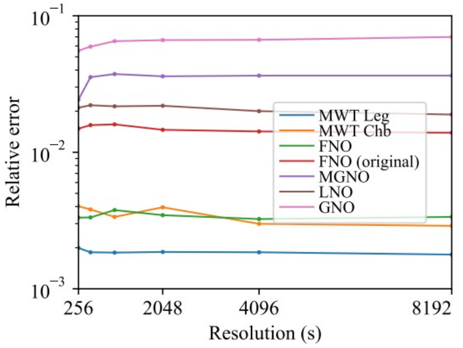


Figure 6: Burgers' Equation validation at various input resolution s. Our methods: MWT Leg, Chb.

Networks	s = 32	s = 64	s = 128	s = 256	s=512
MWT Leg	**0.0152**	**0.00899**	**0.00747**	**0.00722**	**0.00654**
MWT Chb	0.0174	0.0108	0.00872	0.00892	0.00891
MWT Rnd	0.2435	0.2434	0.2434	0.2431	0.2432
FNO	0.0177	0.0121	0.0111	0.0107	0.0106
MGNO	0.0501	0.0519	0.0547	0.0542	-
LNO	0.0524	0.0457	0.0453	0.0428	-

Table 2: Benchmarks on Darcy Flow equation at various input resolution s. Top: Our methods. MWT Rnd instantiate random entries of the filter matrices in (6)-(9). Bottom: prior works on Neural operator.

Networks	$\nu = 1e-3$ $T = 50$ $N = 1000$	$\nu = 1e-4$ $T = 30$ $N = 1000$	$\nu = 1e{-}4$ $T = 30$ $N = 10000$	$\nu = 1e-5$ $T = 20$ $N = 1000$
MWT Leg	**0.00625**	**0.1518**	**0.0667**	**0.1541**
MWT Chb	0.00720	0.1574	0.0720	0.1667
FNO-3D	0.0086	0.1918	0.0820	0.1893
FNO-2D	0.0128	0.1559	0.0973	0.1556
U-Net	0.0245	0.2051	0.1190	0.1982
TF-Net	0.0225	0.2253	0.1168	0.2268
Res-Net	0.0701	0.2871	0.2311	0.2753

Table 3: Navier-Stokes Equation validation at various viscosities ν. Top: Our methods. Bottom: previous works of Neural operators and other deep learning models.

Conclusion & Future work

- Conclusion
  - Another method to approximate the kernel integral
- Future work
  - Different transformation to compute integral operator
  - Different architecture