8th OKUCC

CFD + Neural Network

-Physics informed PDE modelling using Machine learning

2019.09.26
서인덕
NEXTFoam Co., Ltd.

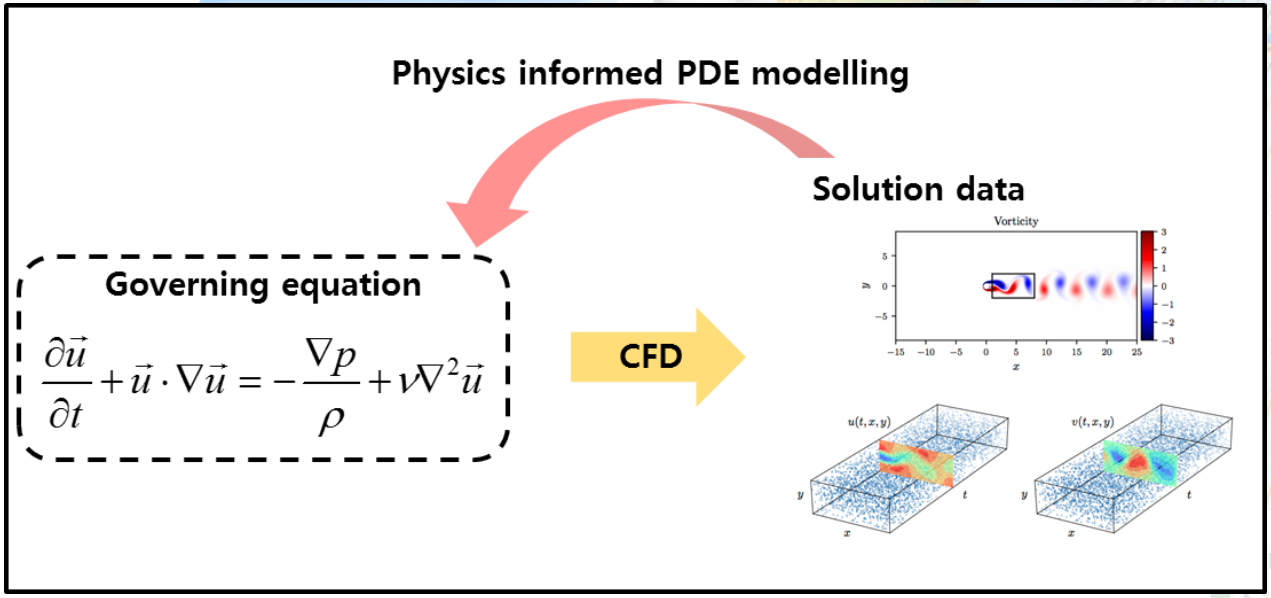
ML를 이용한 Physics informed PDE modelling

✓ Physics informed PDE modelling

- Data-driven solution, Data-driven modelling
- 어떤 현상에 대해 Data만 주어질 경우 역으로 governing equation의 방정식을 modelling 하는 기법
- Machine Learning을 이용하여 unknown PDE 방정식을 구하는 것이 연구의 목표

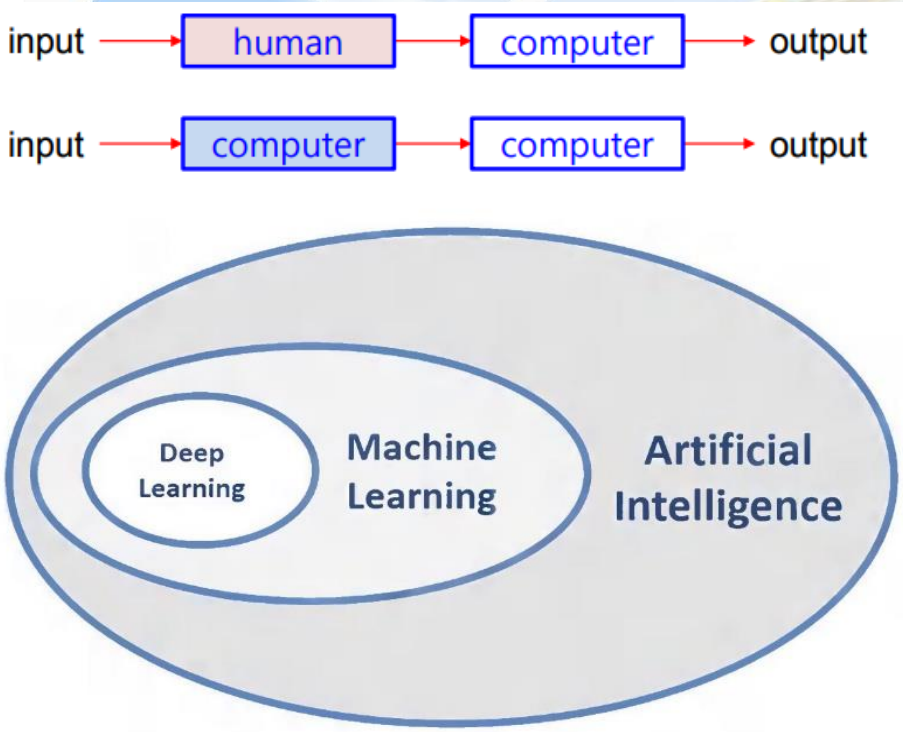
$$\frac{\partial \alpha}{\partial t} + f(t, \vec{x}, etc) = 0$$

- 간단한 Scalar transport 현상에 대하여 Physics informed PDE modelling을 수행 (VOF)
- Machine learning 알고리즘으로는 인공 신경회로망 (Artificial Neural Network, NN) 사용
- 결과 발표 보다는 '삽질' session에 알맞게 study를 진행하면서 어려웠던 점에 대하여 발표할 예정



ML를 이용한 Physics informed PDE modelling

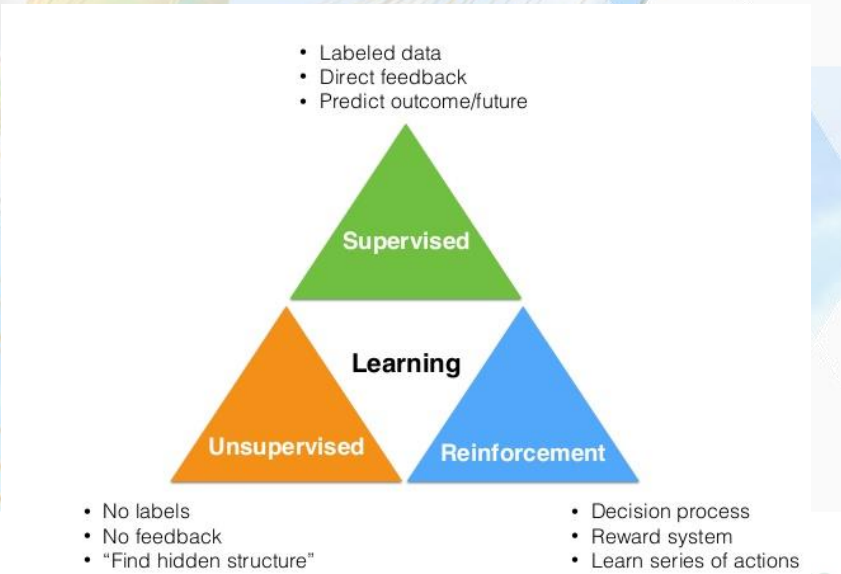
- ✓ **ML(Deep learning)은 AI (Artificial Intelligenc) 분야의 일부**
  - ➢ AI (Artificial Intelligence): system에 의해 만들어진 지능
  - ➢ ML (Machine Learning): 데이터를 이용해서 컴퓨터가 어떤 지식이나 패턴을 학습하는 것
    - "어떤 문제 T(Task)에 관련된 경험 E(Experience)로부터 성과 측정 지표 P(Performance Measure)를 가지고 학습을 진행하는 컴퓨터 프로그램"
  - ➢ DL (Deep Learning): ML의 일종으로 Feature extraction 을 사람이 처리하지 않고 자동으로 추출하는 점에서 차이가 남



ML를 이용한 Physics informed PDE modelling

✓ Learning(학습) 종류

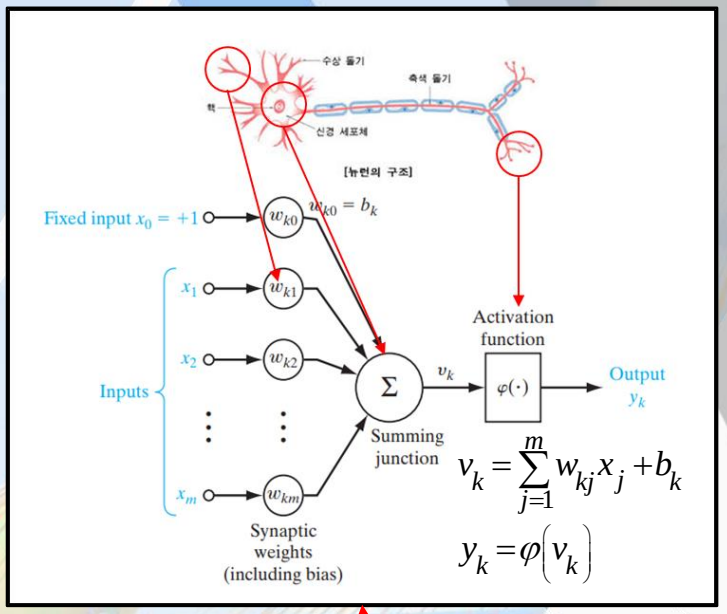
- Learning 의미: 구체적인 task가 주어졌을 때 그 결과를 정량적으로 측정할 수 있고 그 측정 값이 지속적으로 향상될 경우 '학습' 되었다고 정의
- Supervised learning (지도학습)
  - Data의 input 마다 정답에 해당하는 output(label) 이 정해져 있고 준비된 data의 input을 넣었을 때 정해진 output 결과를 추종할 수 있도록 학습시키는 방법 (주입식 교육)
- Un-supervised learning(비지도 학습)
  - Data의 label 이 없음. 대신 현재 보유하고 있는 Data의 구조를 파악하는 사용됨
  - 예) 안면 분류 알고리즘 (남자/여자, 서양/동양, 잘생김/못생김)
- Reinforcement learning(강화 학습)
  - 주변 환경과 interaction을 통해 학습을 진행
  - 특정 action (행동) 마다 grade 가 정해져 있어 높은 grade의 결과를 내놓도록 학습
  - 주로 게임, 자율 주행 자동차와 같은 분야에서 사용됨

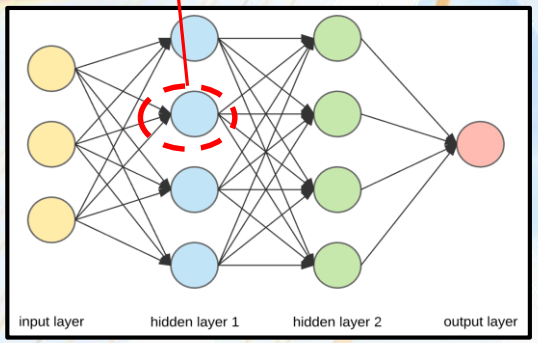


ML를 이용한 Physics informed PDE modelling

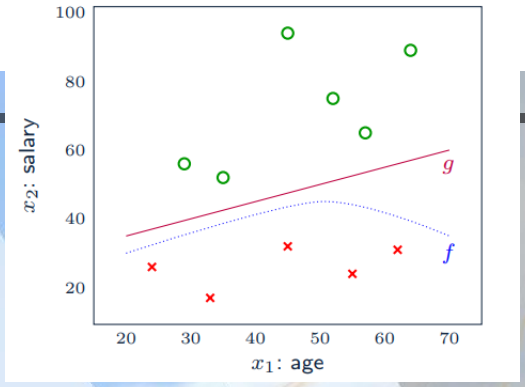
✓ Artificial Neural Network (인공신경망)

- Input layer, 다수의 hidden layer 그리고 output layer 로 구성되어있음
- 각 layer는 다수의 Perceptron으로 이루어져 있음
- 각각의 Perceptron 은 인간 뇌세포의 뉴런으로 생각하면 됨
  - Input → 수상돌기에서의 전기 자극
  - Summing junction → 신경 세포체
  - Activation function → 역치
  - Output → 축색 돌기 말단
- 각 NN 에서 하나의 perceptron은 다수의 input을 받으며 다른 perceptron으로 하나의 output을 보냄
  - Input value에 weight(w)를 곱한 후 bias(b)를 더함위해서 구한 결과를
  - 일반적으로 logits 라고 하는데
    이에 Activation function을 곱해 output으로 전달
- NN 에서 학습은 모든 input 에 대한 output 이 Training data의 값과의 차이가 작아지도록 모든 perceptron의 weight 와 bias 를 최적화 시키는 것



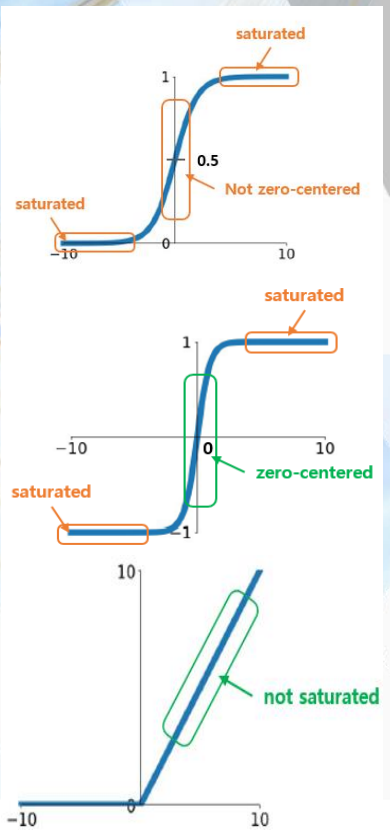


ML를 이용한 Physics informed PDE modelling



✓ Artificial Neural Network (인공신경망)

- ➢ 하나의 Perceptron 은 input을 linear 하게 조합하여 하나의 output 을 계산
  - Linear combination 이기 때문에 단일 perceptron 은 linear한 결과만 도출
  - Nonlinear 한 결과를 얻기 위해선 과거에는 Kernal 을 곱하여 nonlinear한 결과를 얻음
  - Activation function 이 존재하여 non-linearity 확보
- ➢ Activation function
  - Simoid: $\sigma(X) = \frac{1}{1+e^{\{-x\}}}$
  - 입력이 크게 될 경우 Gradient 0으로 수렴하여 Vanishing gradient 발생
  - 평균값이 0이 아니라 편향 이동(bias shift) 현상 발생, 출력 가중치 합이 입력 가중치 합보다 커져 계속 분산이 커져 data 손실이 발생하는 현상
  - Tanh: $\sigma(X) = \frac{1-e^{-x}}{1+e^{-x}}$
  - Vanishing gradient 현상이 발생
  - 편향 이동 발생 안함
  - RELU(Rectified Linear Unit): $\sigma(X) = \max(0, x)$
  - 빠른 계산속도. Vanishing gradient 현상을 개선할 수 있음
  - Dead RELU (학습 도중 대부분 뉴런의 값이 0으로 유지되어 아무 역할도 하지 않는 현상)
  - Dead RELU 현상을 해결하기 위해 RELU family (Leaky RELU, PRELU 및 ELU) 가 존재



ML를 이용한 Physics informed PDE modelling

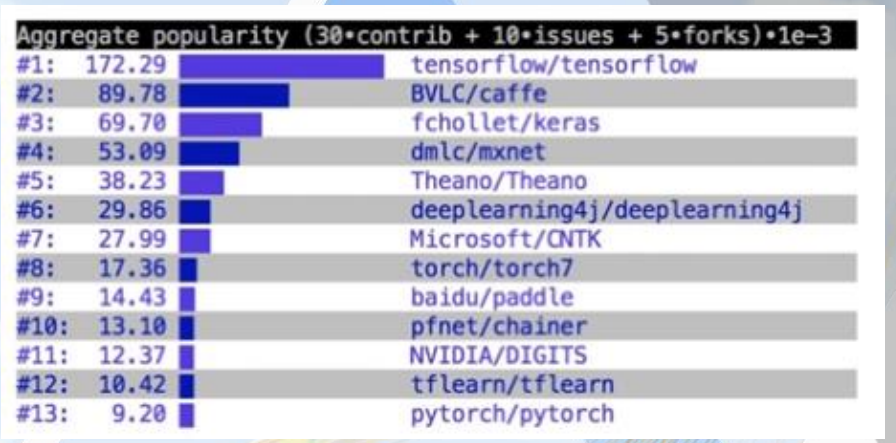
- ✓ **Artificial Neural Network (인공신경망)**
  - ➢ 신경회로망 학습 순서

1. Training data 상에서 input data $\tilde{x}$ 와 output data $\tilde{y}$ 를 준비
2. Neural Network 구성 및 초기화
   - NN에 들어가게 되는 Input feature 결정
   - Inputlayer, Hidden layer, output layer 개수 결정
   - NN weight 및 bias 초기화 (random normal)
3. Input data 를 이용하여 feature 추출
4. Feature 를 NN 에 input으로 가하여 NN output 계산
5. NN output 과 Training data 사이의 Error 산출 (Mean-Square Error, Cross Entropy error 등등)
6. Back propagation 을 이용하여 Weight 대비 Error gradient 계산
7. Optimizer 를 이용하여 weight update
8. 위의 과정 반복

ML를 이용한 Physics informed PDE modelling

✓ **TensorFlow**

- ➢ 가장 대중화 되어있는 Deep learning 프레임워크
- ➢ Python script
- ➢ 사용자 Interface를 강화한 Keras 존재
- ➢ GPU version 설치……



Theano	- 캐나다 몬트리올 대학 - Python - 오픈소스 - Convolution Neural Network 및 Auto-Encoder 에 최적화 - API가 강화된 상위 버전 프레임 워크가 다수 존재 - 에러 log가 부족 - 규모가 큰 모델에는 계산 속도가 느림 - 사전 학습 모델 지원 불충분 - 구조가 복잡	**Torch**	- 뉴욕 대학교에서 개발 - Lua script language 사용 - 오픈 소스 - C/C++ library 사용 (연산 속도 빠름) - 페이스북 인공지능 연구소 및 구글 딥마인드에서 사용되고 있음 - Lua script language 의 경우 Java script와 비슷 (쉽고 직관적) - Lua script language 커뮤니티 부족
Caffe	- 캘리포니아 버클리 대학 - C/C++/python interface - 오픈소스 - 산업에서 일부 사용하고 되어있음 - Computer vision 에 특화 - Community 활동이 왕성함 - Update 가 느림 - 최신 버전 GPU 호환성 문제 및 최적화가 느림 - 병렬 계산 API 부족	**TensorFlow**	- 구글 브레인 프로젝트 팀에서 개발 - 오픈소스 라이선스 제공 - C/C++ 기반 엔진 python interface - Neural Network 및 reinforcement learning 지원 - GPU 활용에 최적화 - Parallel computing 지원 (구글 cloud 서비스에서만) - Tensor-board (시각화 도구 제공) - 사전학습 모델이 많이 제공되지 않음 - GPU version 설치 …

ML를 이용한 Physics informed PDE modelling

✓ Physics informed PDE Modelling (1D case, 열전도 문제)

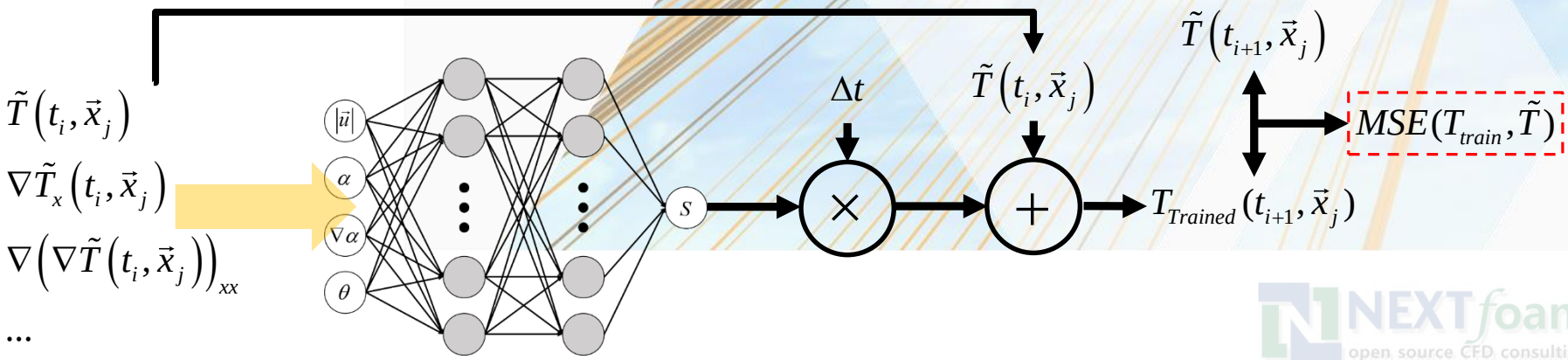
- ➢ 우리가 모르는 다음과 같은 Governing equation 이 존재한다고 가정
  - Governing equation 은 모르지만 실험 혹은 다른 방법으로 통해 실제 현상의 온도 분포 data가 존재
  - 이 data를 Training data 로 선정

$$\frac{\partial T}{\partial t} + 0.01\nabla^2 T = 0$$

- ➢ Governing equation 에서 일부분을 unknown term 으로 가정 후 Neural Network 가 학습 되어 unknown term을 표현 하도록 가정

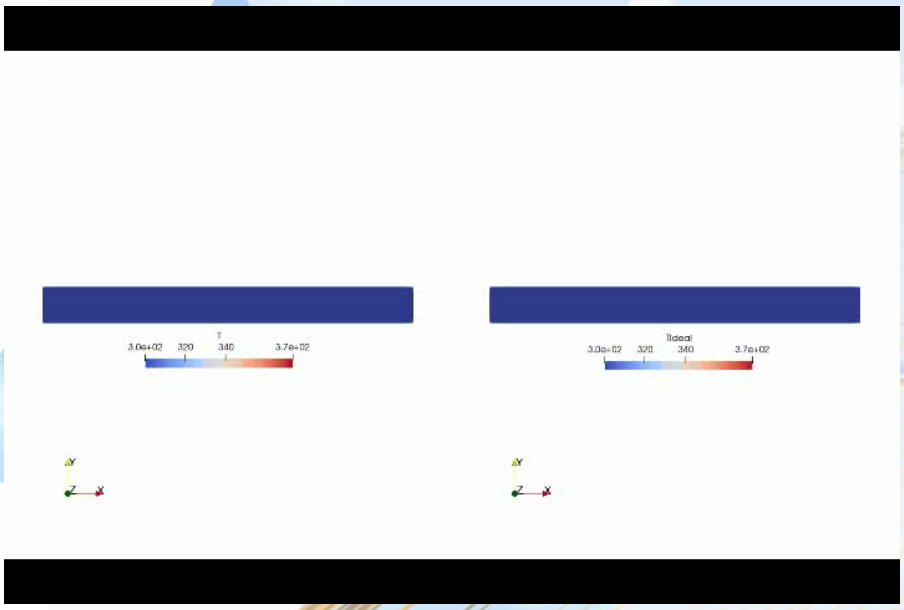
$$\frac{\partial T}{\partial t} + NN(t, x, T, \ldots) = 0$$

- ➢ 임의의 시간 t_i , 위치 x_j 에 대해 feature 추출
  - Second order gradient (Laplacian 아님) 까지 사용
  - Backpropagation, AdamOptimizer 를 이용하여 weight update
  - 모든 cell, 모든 time 에서 값을 사용 $T\left(t_i, \vec{x}_j\right), \nabla T_x\left(t_i, \vec{x}_j\right), \nabla\left(\nabla T\left(t_i, \vec{x}_j\right)\right)_{xx} \ldots$



ML를 이용한 Physics informed PDE modelling

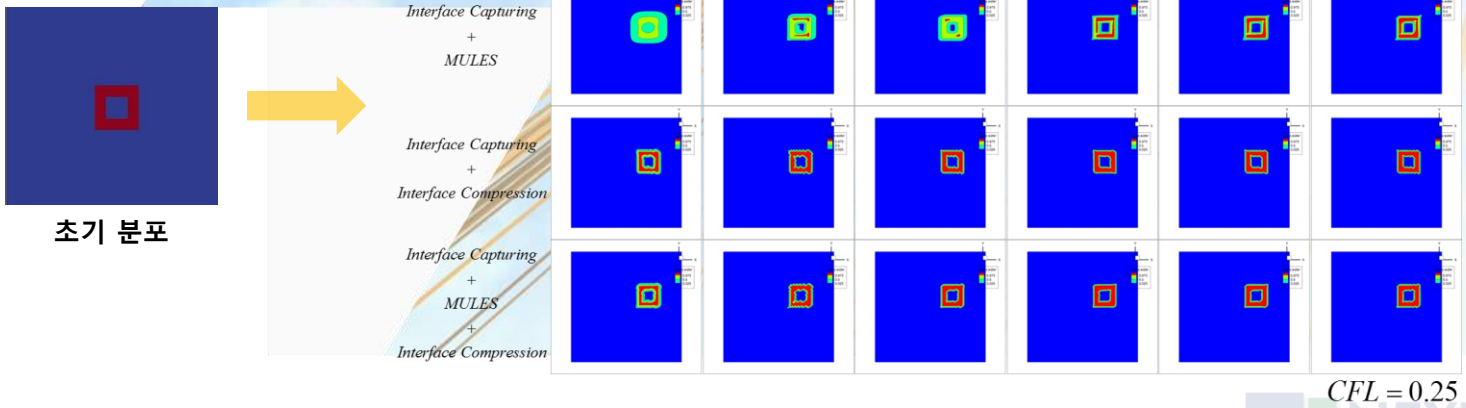
- ✓ **Physics informed PDE Modelling (1D case, 열전도 문제)**
  - ➢ 간단하게 구성하여 가능성 확인
  - ➢ 약간의 오차가 존재하지만 어느정도 일치함 확인



ML를 이용한 Physics informed PDE modelling

✓ Physics informed PDE Modelling (2D case, VOF compression term)

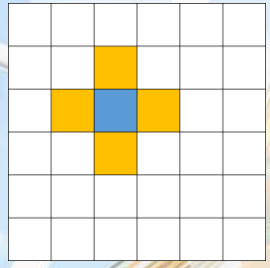
- ➢ Rudman test (2차원 VOF bench mark test 일종)
  - 전체 2차원 영역에 일정한 속도
  - VOF 방정식만 해석 → 제대로 된 VoF transport equation 을 사용할 경우 기존 형상을 최대한 유지한채로 α (VOF) 분포가 평행이동 해야함
  - Numerical Diffusion 및 유계성 문제 (wiggle, interface 가 찌그러짐) 가 없는 결과가 목표
  - Interface capturing method (CICSAM, HRIC, MHRIC), Interface compression method 그리고 Flux Correction Transport (FCT, OpenFOAM 에선 MULES) 에 대해 비교

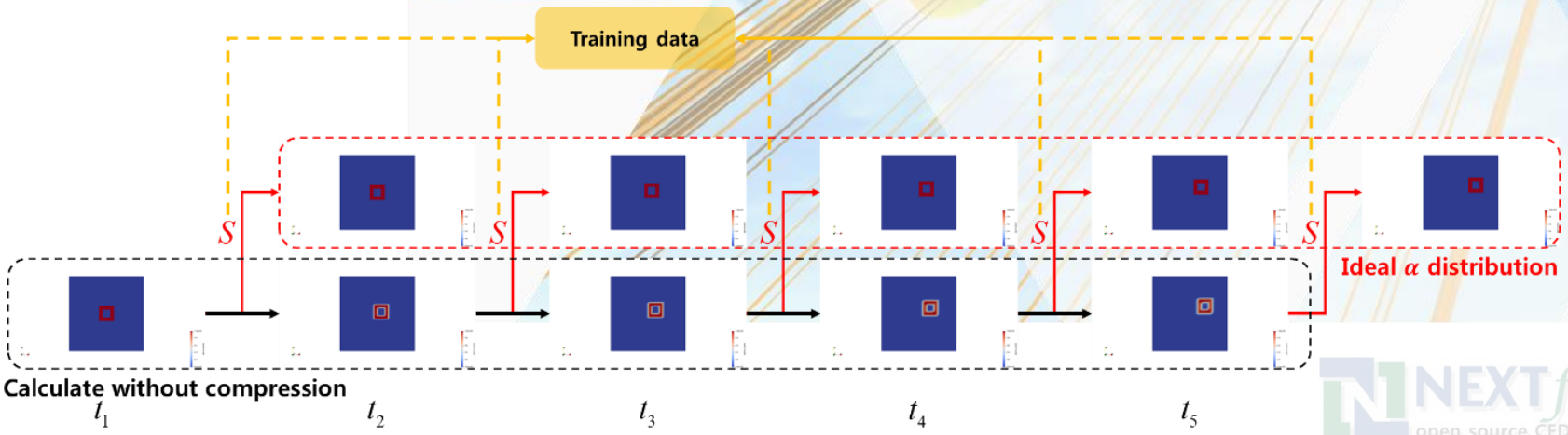


ML를 이용한 Physics informed PDE modelling

✓ Physics informed PDE Modelling (2D case, VOF compression term)

- ➢ Rudman test (2차원 VOF bench mark test 일종)
  - 전체 PDE 학습하기 이전에 OpenFOAM 에서 사용하는 interfaceCompression term 학습을 목표로 함
  - Interface-compression 을 적용한 case 와 그렇지 않은 경우의 PDE 방정식의 차이 (source term을 추출 후 학습 시킴)
  - $\alpha, \nabla\alpha_x, \nabla\alpha_y, \vec{u}$ 사용 (실제 interface compression 알고리즘 상에서 $\nabla\alpha$ 와 $\vec{u}$ 가 이루는 각이 중요)
  - 다만 주위 인접한 cell 중심에서의 값을 포함하여 진행

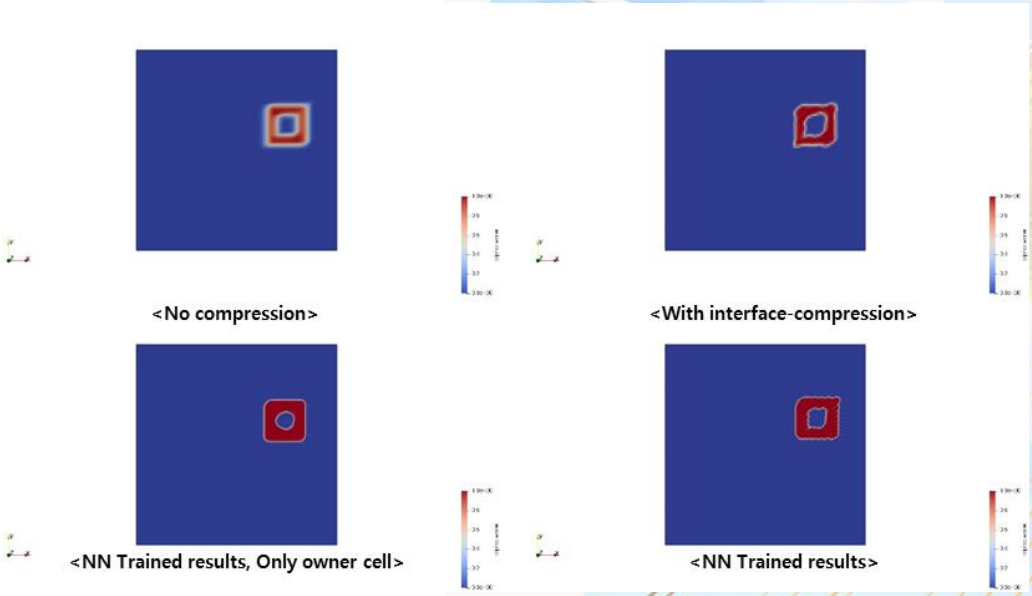


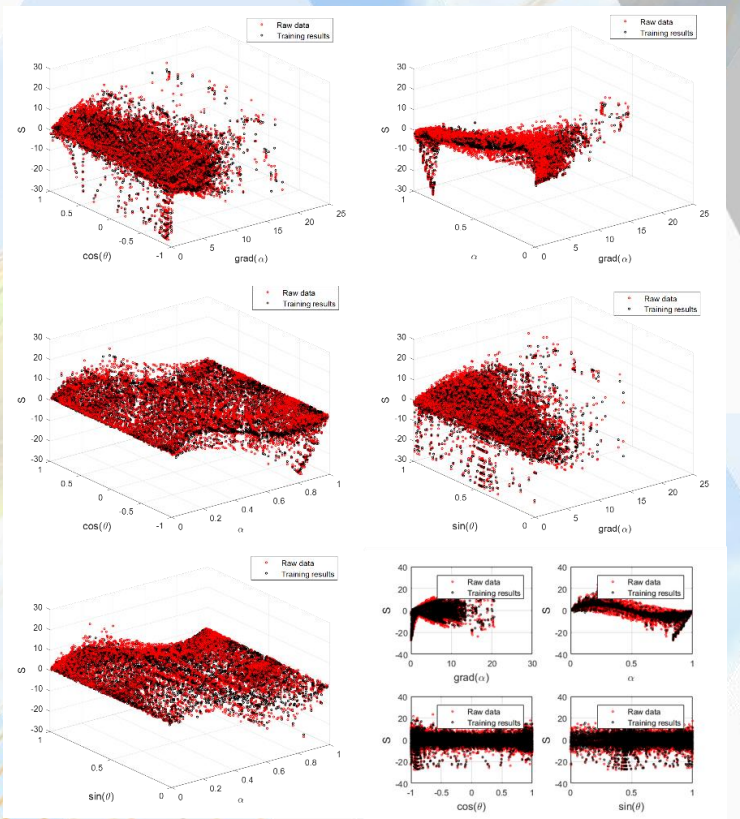


ML를 이용한 Physics informed PDE modelling

✓ Physics informed PDE Modelling (2D case, VOF compression term)

- ➢ Rudman test (2차원 VOF bench mark test 일종)
  - 추출 된 source term 들을 학습된 source term들과 비교
  - Input 을 포함한 hyper plane 상에서 표시
    적색: Trained interface-compression term
    흑색: Weller's interface-compression term)
  - 어느정도 비슷한 경향이 나타나는 것을 확인



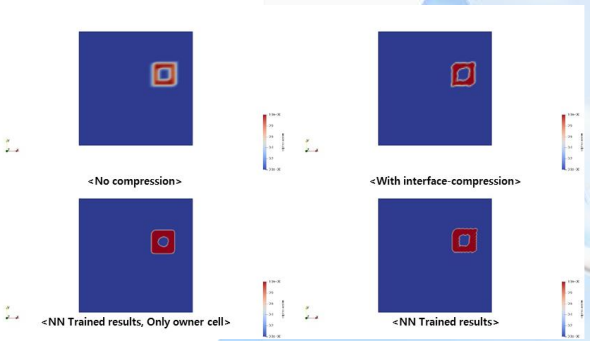


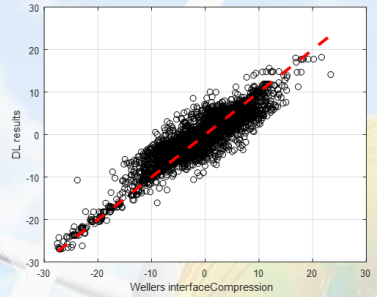
ML를 이용한 Physics informed PDE modelling

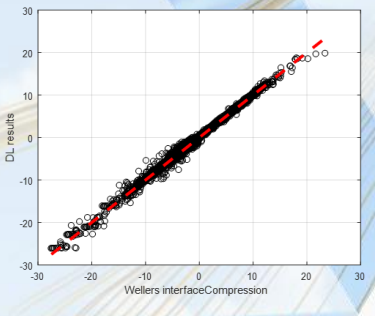
✓ Physics informed PDE Modelling (2D case, VOF compression term)

➢ Rudman test (2차원 VOF bench mark test 일종)

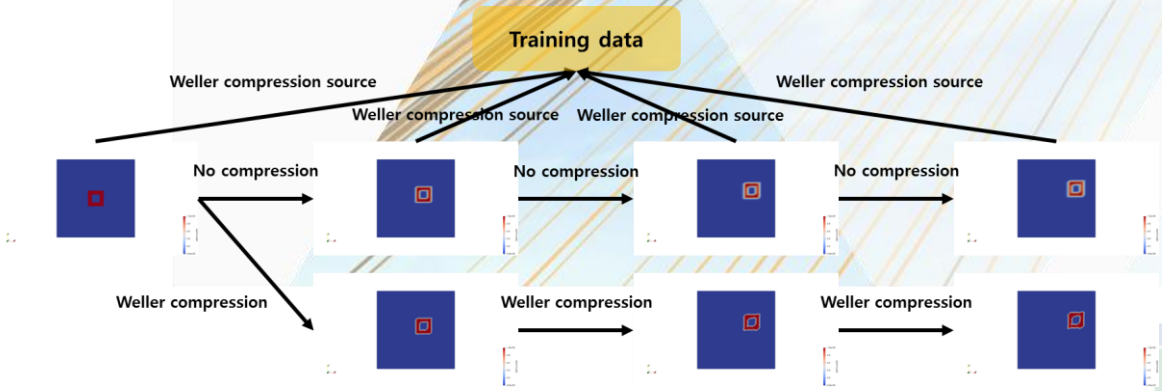
- Owner cell 의 물리량만 input으로 사용했을 경우보다 adjacent cell 의 물리량도 input에 포함했을 경우 실제 Training 에 사용한 interface-compression term과의 상관도가 높음







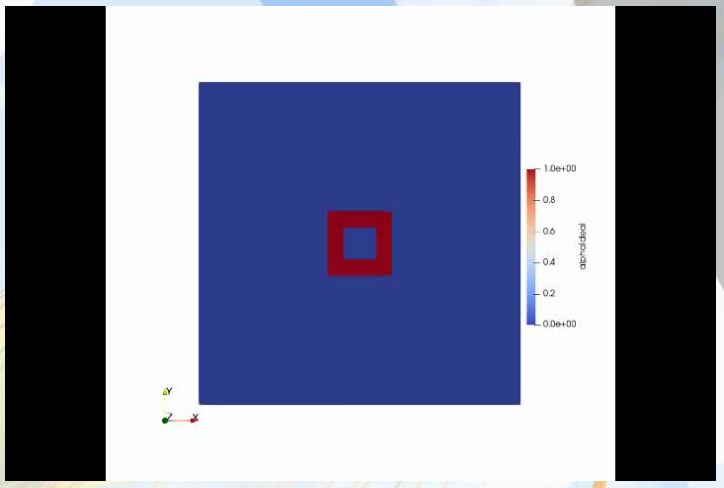
- Training 에 사용한 compression 결과와 NN으로 학습한 결과의 차이가 발생하는 이유
  - Compression 을 적용 안한 상태에서 compression term을 추출하여 Training 을 하였기 때문에
  - Compression 을 적용한 상태에서 compression term을 추출하였을 경우는 동일하게 나올 것으로 추정

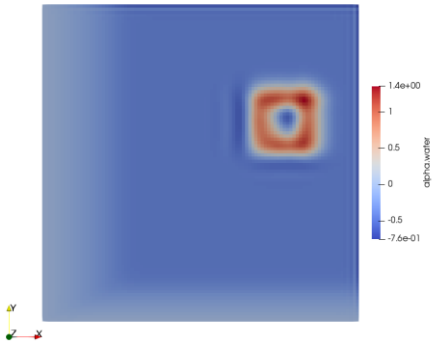


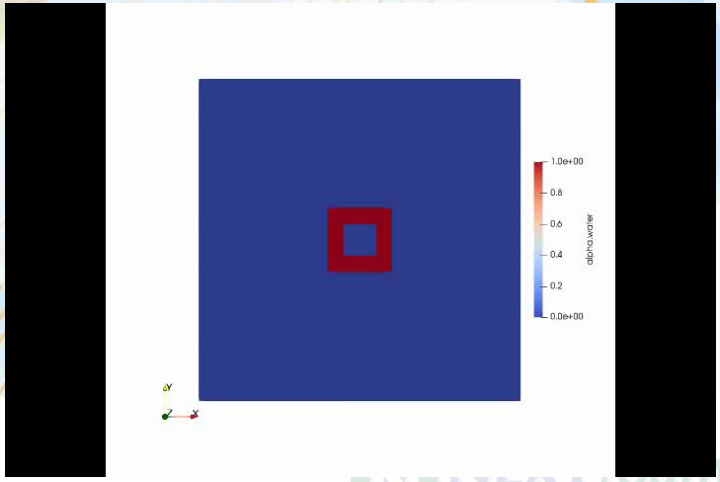
ML를 이용한 Physics informed PDE modelling

- ✓ **Physics informed PDE Modelling (2D case, VOF transport equation)**
  - ➢ 이상적인 VOF 분포를 이용하여 VOF 수송 방정식을 모델링 하는 것을 목표로 한다
    - 1D 열전도 학습과 동일한 방식으로 학습 진행
  - ➢ Informed data
    - OpenFOAM 상에서 interpolation 하여 구함
  - ➢ Training input data
    - 더 고차 gradient 값은 계산 용량을 감당할 수 없음
    - Linear gradient 만 사용할 경우 유계성이 좋지 않아 wiggle 이 나타남

$$\alpha, \left(\frac{\partial \alpha}{\partial x}, \frac{\partial \alpha}{\partial y}, \frac{\partial^2 \alpha}{\partial x \partial x}, \frac{\partial^2 \alpha}{\partial x \partial y}, \frac{\partial^2 \alpha}{\partial y \partial y} \right)_{linear}, \left(\frac{\partial \alpha}{\partial x}, \frac{\partial \alpha}{\partial y}, \frac{\partial^2 \alpha}{\partial x \partial x}, \frac{\partial^2 \alpha}{\partial x \partial y}, \frac{\partial^2 \alpha}{\partial y \partial y} \right)_{upwind}$$



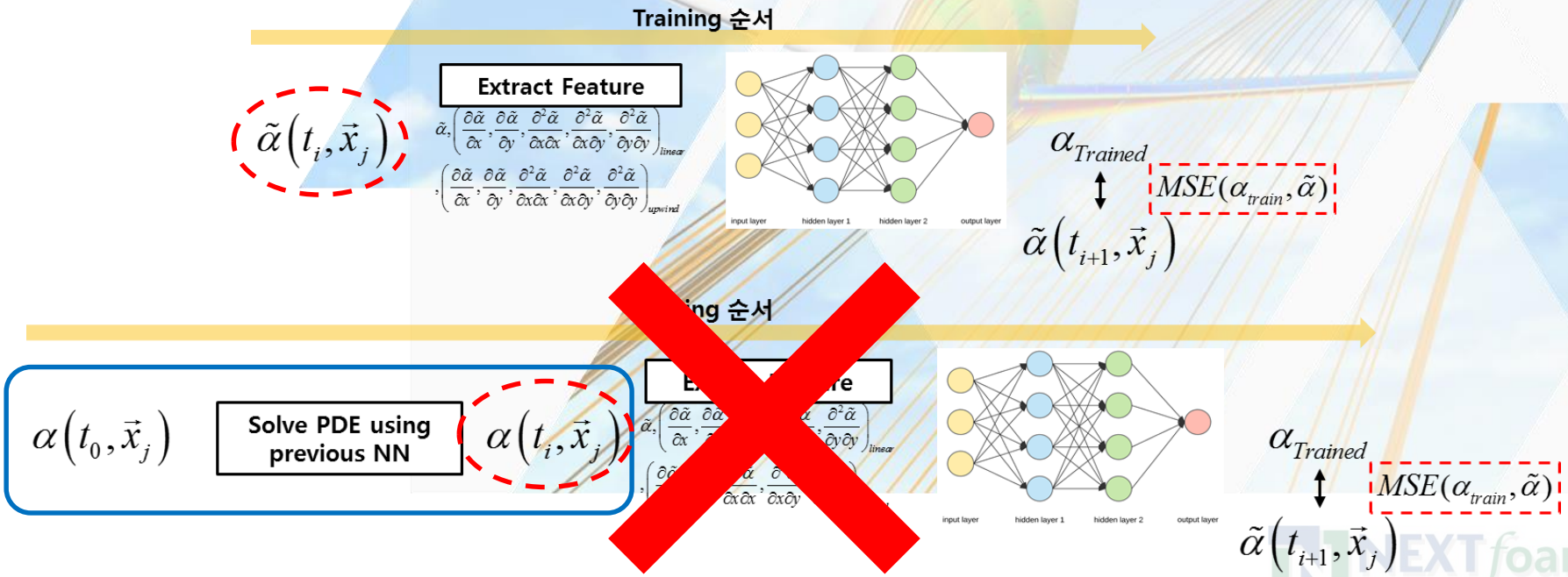




ML를 이용한 Physics informed PDE modelling

✓ Physics informed PDE Modelling (2D case, VOF transport equation)

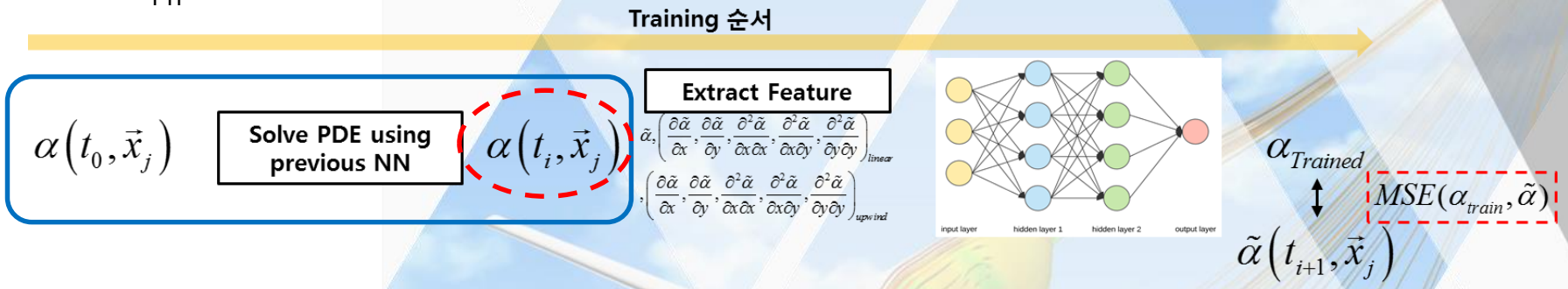
- ➢ 왜 정확한 결과가 나타나지 않을까?
  - 1D 열전도 문제의 경우 매 time step 마다 열이 x방향으로 전달되면서 변하기 때문에 Training data에서 겹치는 경우가 거의 발생하지 않음
  - 반면 Rudman test의 경우 대부분의 data가 의미가 없음 (background는 α 값이 0)
  - Unsteady 문제이지만 moving coordinate 상에서는 steady 문제이기 때문에 Training data가 많다고 하더라도 겹치는 data가 대부분임
- ➢ 개선 방법: Informed Data 만을 가지고 training 을 하지 말고 해석을 진행하면서 training을 병행



ML를 이용한 Physics informed PDE modelling

✓ Physics informed PDE Modelling (2D case, VOF transport equation)

➢ Training data가 아닌 이전 step (epoch) NN 결과를 이용하여 CFD 해석한 값을 이용한 방법이 실패한 이유

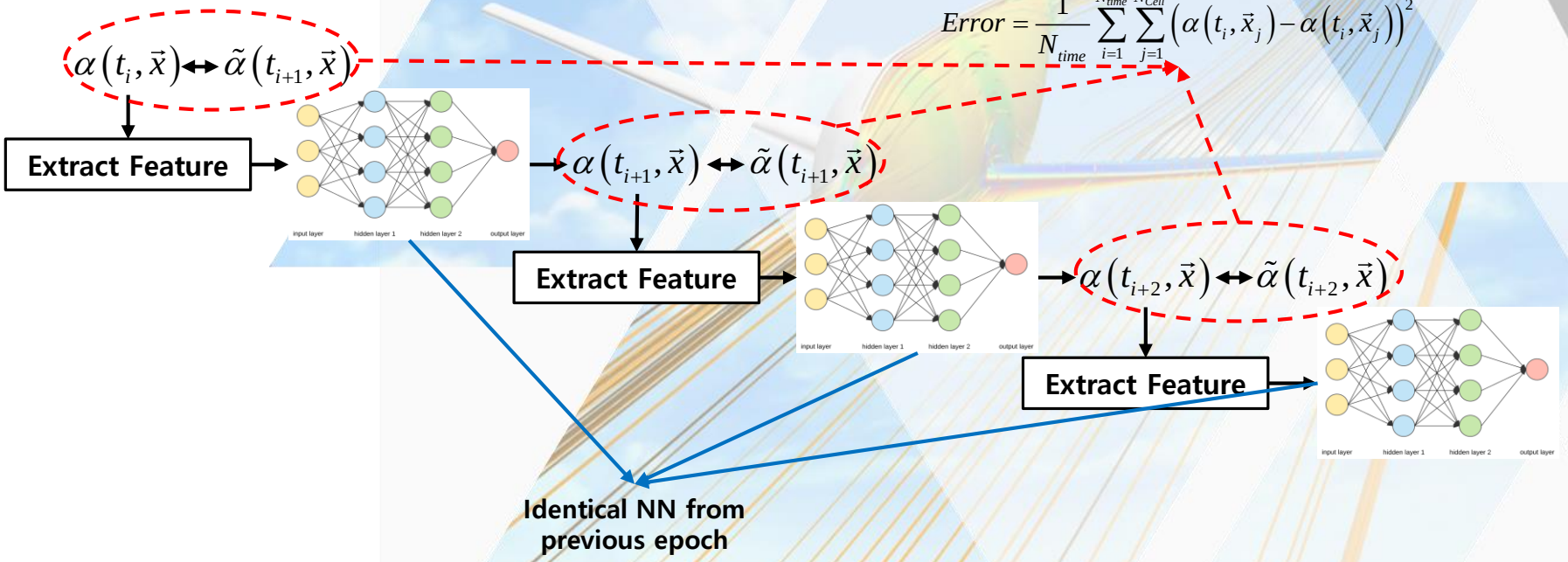


- Training 이란 결국 Neural Network 상의 weight 및 bias들을 update 하는 과정
- 사용자가 지정한 error를 최소화하는 극점을 찾음 (gradient descent 방식을 사용)
  - Backpropagation을 이용하여 gradient 를 계산하고 optimizer를 이용하여 gradient descent 방식을 수행
- 따라서 학습 초기에는 Neural network 의 weight 및 bias가 터무니 없는 값을 갖음
  - 이 값을 이용하여 CFD 해석을 수행하면 발산
- 초기 weight 및 bias 가 어느정도 의미 있는 값을 갖더라도 △t 단위의 값을 이용하여 학습하는 것에서 문제가 발생
  - $\alpha(t_n, x)$ 와 $\alpha(t_m, x)$ (n<m) 은 같은 NN 결과를 이용하여 CFD 해석을 수행한 결과이기 때문에 종속관계임
  - 하지만 위와 같은 방식은 두 값에서 추출한 Feature 가 독립적으로 가정하고 학습을 수행한다
  - 일관성 없는 training data set을 만들어 버려서 제대로 under fitting 된 학습 결과를 초래함

ML를 이용한 Physics informed PDE modelling

✓ Physics informed PDE Modelling (2D case, VOF transport equation)

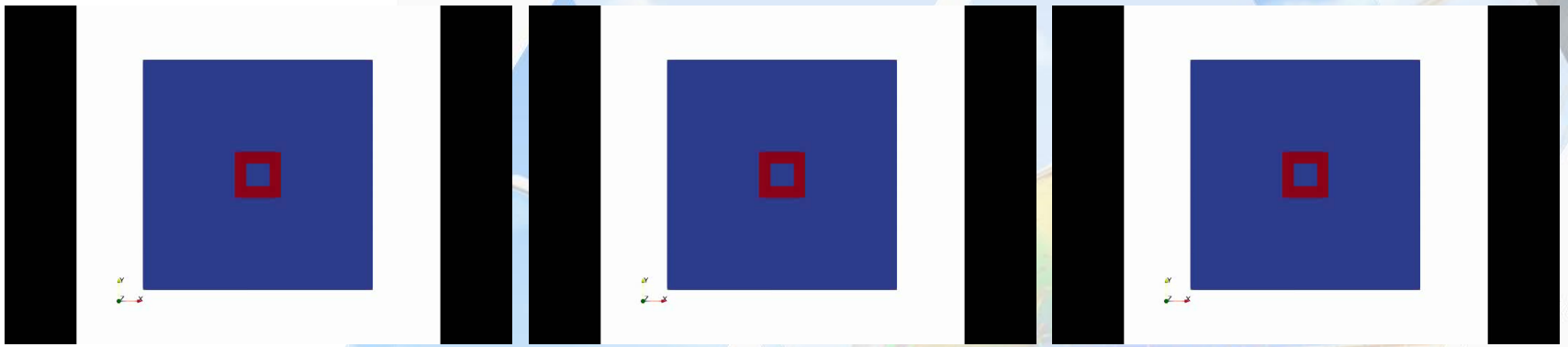
- ➢ Multi Δt block training
  - 앞서 설명한 방식은 하나의 Δt 사이에서 input과 output 을 이용하여 학습을 수행
  - Multi Δt block training 에서는 여러 time step 동안의 값을 이용하여 학습을 수행
  - 하나의 cell 단위에서 학습을 수행하는 것이 아니라 해석 영역 전체를 하나의 data set으로 간주
  - 이전에는 각 cell 단위도 연관이 없다고 가정하고 하나의 training set 으로 구성



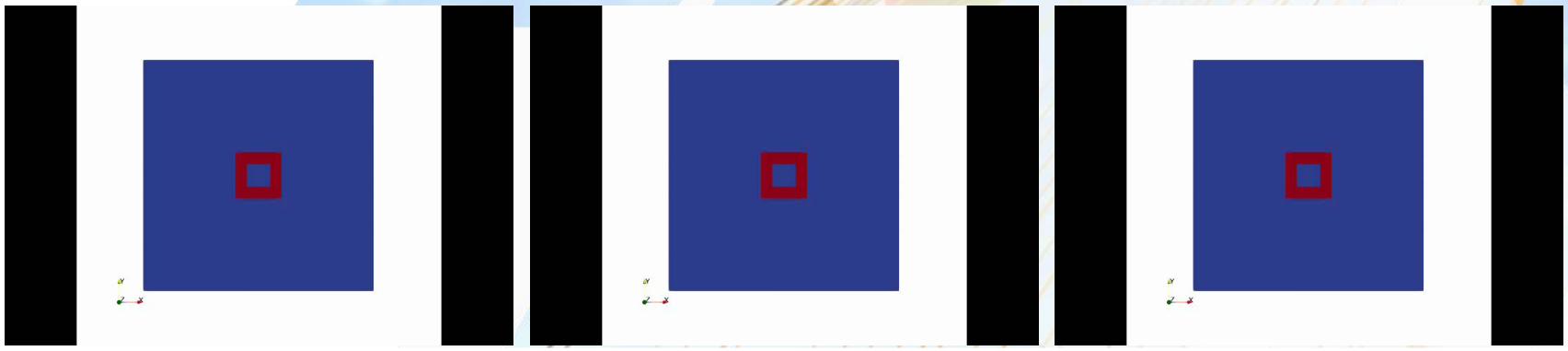
ML를 이용한 Physics informed PDE modelling

✓ Physics informed PDE Modelling (2D case, VOF transport equation)

- ➢ Multi Δt block training
  - Δt block 개수가 클수록 더 좋은 결과를 도출



<1Block results>　　<5Block results>　　<10Block results>

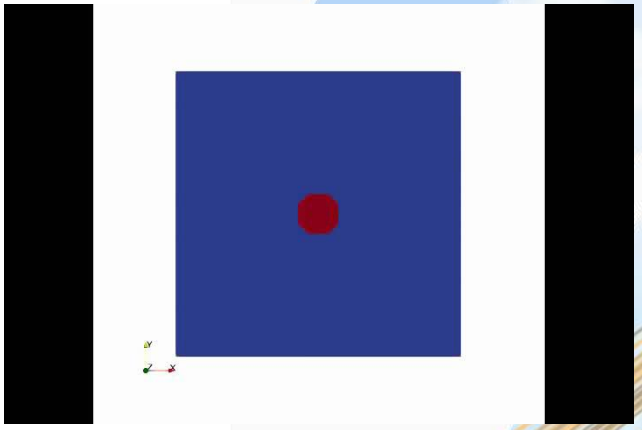


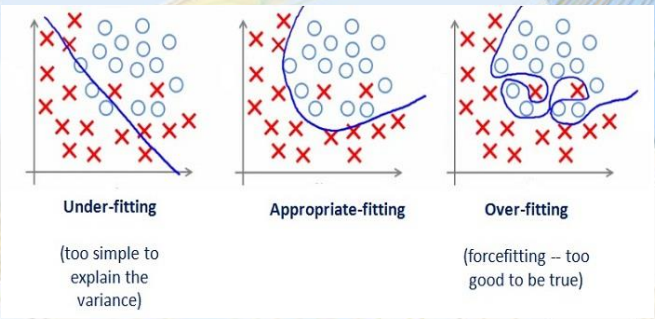
<20Block results>　　<40Block results>　　<80Block results>

ML를 이용한 Physics informed PDE modelling

✓ Physics informed PDE Modelling (2D case, VOF transport equation)

- ➢ Multi Δt block training
  - 학습을 진행할 때 Overfitting 이 되어 training case 에서는 좋은 결과를 내지만 다른 case에서는 나지 않는 경우가 발생
  - 확인하기 위해 다른 초기조건에 대해 test 수행

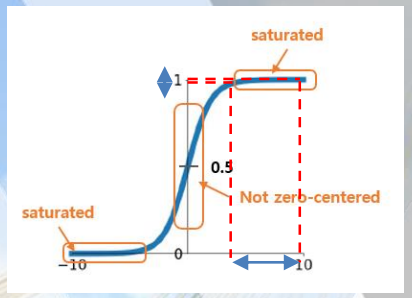




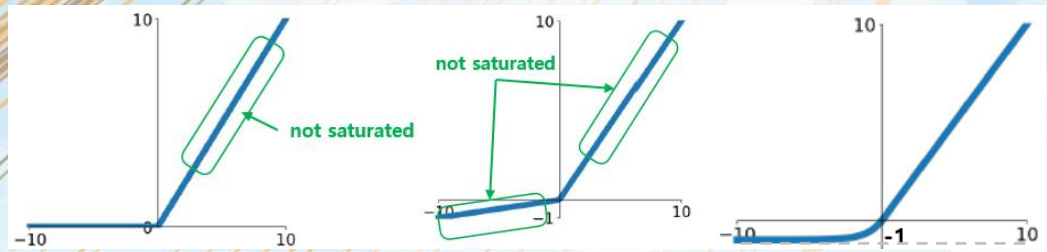
ML를 이용한 Physics informed PDE modelling

✓ 학습 결과 talk

➢ Activation function 관련 issue



- Sigmoid function 및 tangent hyperbolic: Vanishing gradient 현상이 문제가 됨
- Hidden layer 개수가 늘어나지 않더라도 문제가 발생
  - Input signal의 크기가 크게 차이가 나도 output signal 의 큰 차이가 발생하지 않아 data 손실이 발생
  - 수치 해석 상에서 발생하는 오차가 weight update 할 때 큰 영향을 줄 수 가 있음
- ReLU family activation function 을 사용하면 해결 가능
- ReLU: dead ReLU 현상이 발생함
  - Classification 과 같은 문제에서는 ReLU와 같이 output value 가 0으로 가는 것이 non-linearity 에 도움이 되지만 수치해석 분야에서는 value 값이 극단적으로 0이 되는 경우 결과의 연속성에 방해가 됨
- Leaky-ReLU: 수렴속도 개선에 효과적임
  - Input signal이 0인 위치에서 미분 값이 불연속 적이기 때문에 학습 시 안정적으로 수렴하지 않는 경우가 발생
- ELU: 수렴속도 개선에 효과적임
  - Signal 값이 -1로 고정 되는 경우 문제가 될 수 있음
- 본 연구에서는 Activation function 을 사용하지 않음
  - Input feature들의 non-linearity 확보를 할 수 없음
  - Non-linearity를 확보하고 싶은 경우 기존 방식의 NN 연결이 아닌 customized 된 연결을 고려해야함



ML를 이용한 Physics informed PDE modelling

✓ 학습 결과 talk

➢ Implicit vs Explicit

- Input feature 를 high order gradient 까지 확보할 경우 생각해 보아야함
- Training은 Implicit 한 PDE를 가정하고 모델링을 함
- PDE를 NN로 구성했을 경우 이를 적용하여 해석할 때 Implicit 하게 matrix solve 할 수 없음 따라서 explicit 하게 검증을 수행

➢ Computing power

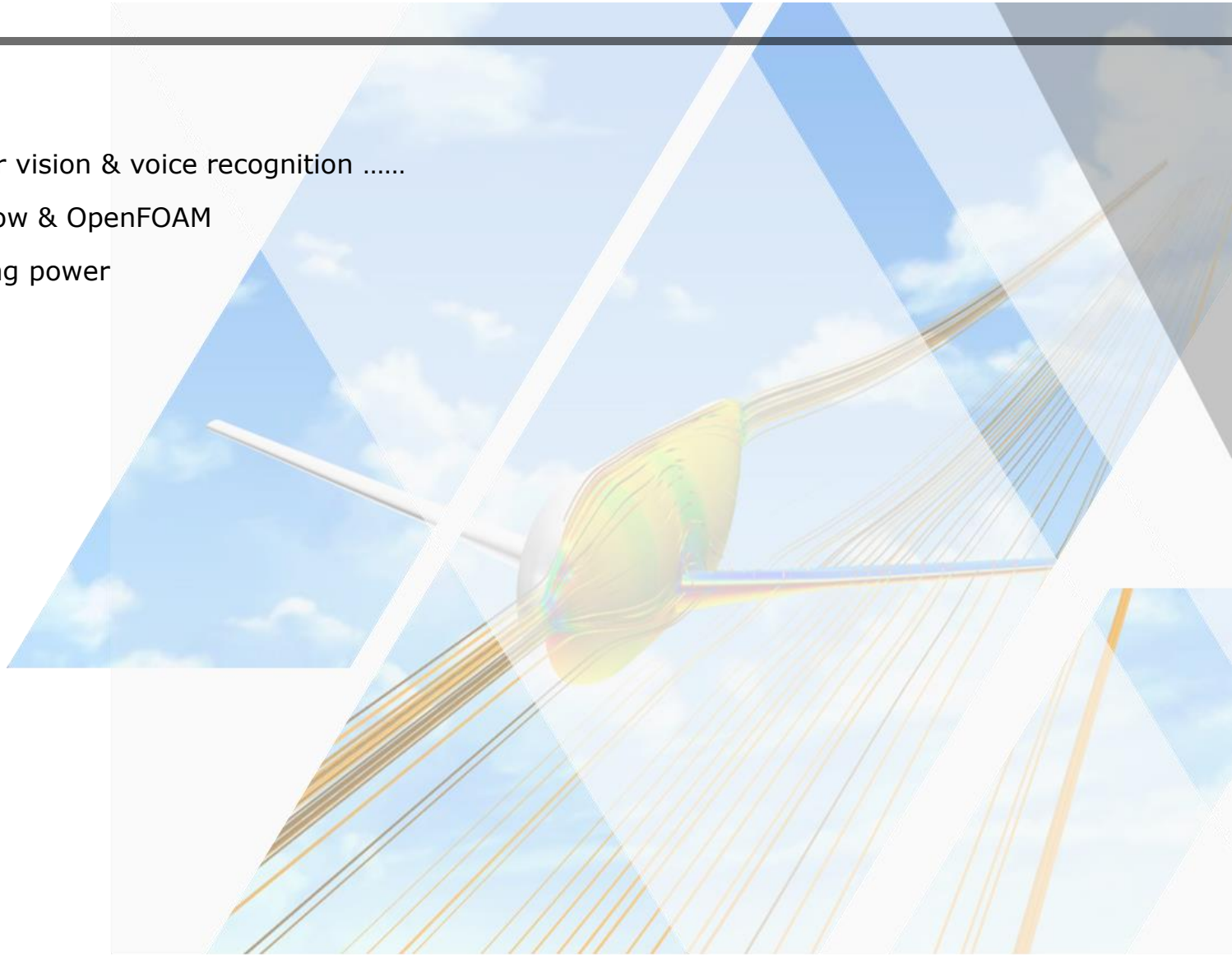
- 현재 모든 cell 을 한번 학습에 모두 사용함
- $N_{cell} \times N_{feature}$ 개수 많금의 input이 필요함
- Delta T block 개수가 늘어날 경우 모든 time step 에서의 input feature가 필요함 $N_{cell} \times N_{feature} \times N_{block}$
  - 많은 memory 가 필요함
  - 현재 개인 PC (GTX 970 Ti) 를 이용하여 학습을 수행함
  - Computing power가 모잘라 충분히 학습을 진행하기 못하여 오차가 발생한 것으로 추정

➢ 가중치 초기화

- NN update 초기 weight 및 bias 가 모두 0이거나 동일한 특정 값을 갖을 경우 학습이 초기 weight update 가 제대로 이루어지지 않음
  - 모든 perceptron 이 동일한 output 을 내기 때문에 NN 이 마치 하나의 perceptron 과 같이 행동하기 때문
- 초기 NN weight 및 bias 의 asymetric 해야지 학습이 효과적으로 진행 됨
  - Asymmetric 할 경우 CFD 해석시 발산하는 결과를 초래할 수 있음

ML를 이용한 Physics informed PDE modelling

- ✓ **애로사항**
  - ➢ Computer vision & voice recognition ……
  - ➢ Tensor-flow & OpenFOAM
  - ➢ Computing power



work 및 Auto-Encoder 에 최적화
프레임 워크가 다수 존재

속도가 느림
충분

되어있음
함

문제 및 최적화가 느림

cs informed PDE mod

d PDE modelling

ution, Data-driven modelling

Data만 주어질 경우 역으로 governing

<No compression>

<NN Trained results, Only owner cell>

<NN Train

GPU version 출시

Physics informed PDE modelling

Training 순서

Solution data

Extract Feature

$$\alpha_{linear}\left(\frac{\partial\tilde{\alpha}}{\partial x},\frac{\partial\tilde{\alpha}}{\partial y},\frac{\partial^2\tilde{\alpha}}{\partial x\partial x},\frac{\partial^2\tilde{\alpha}}{\partial x\partial y},\frac{\partial^2\tilde{\alpha}}{\partial y\partial y}\right)$$

$$\alpha_{upwind}\left(\frac{\partial\tilde{\alpha}}{\partial x},\frac{\partial\tilde{\alpha}}{\partial y},\frac{\partial^2\tilde{\alpha}}{\partial x\partial x},\frac{\partial^2\tilde{\alpha}}{\partial x\partial y},\frac{\partial^2\tilde{\alpha}}{\partial y\partial y}\right)$$

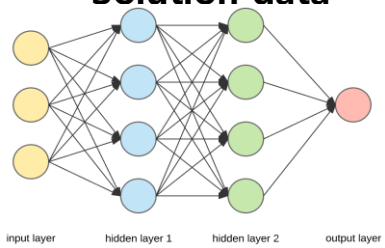
CFD

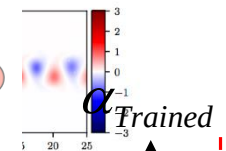
Governing equation

$$\alpha\left(t_i,\vec{x}_j\right)$$

$$\frac{\partial\vec{u}}{\partial t}+\vec{u}\cdot\nabla\vec{u}=-\frac{\nabla p}{\rho}+\nu\nabla^2\vec{u}$$

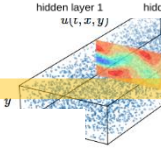
Training 순서



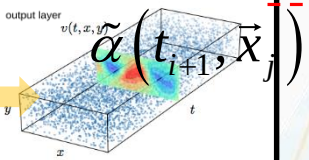


$\alpha_{Trained}$

$MSE(\alpha_{train},\tilde{\alpha})$



$\tilde{\alpha}\left(t_{i+1},\vec{x}_j\right)$

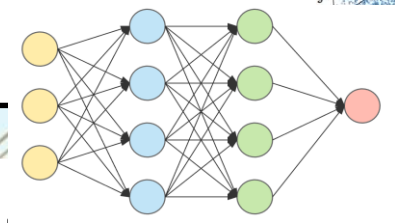


Extract Feature

$$\alpha\left(t_i,\vec{x}_j\right)$$

$$\alpha_{linear}\left(\frac{\partial\tilde{\alpha}}{\partial x},\frac{\partial\tilde{\alpha}}{\partial y},\frac{\partial^2\tilde{\alpha}}{\partial x\partial x},\frac{\partial^2\tilde{\alpha}}{\partial x\partial y},\frac{\partial^2\tilde{\alpha}}{\partial y\partial y}\right)$$

$$\alpha_{upwind}\left(\frac{\partial\tilde{\alpha}}{\partial x},\frac{\partial\tilde{\alpha}}{\partial y},\frac{\partial^2\tilde{\alpha}}{\partial x\partial x},\frac{\partial^2\tilde{\alpha}}{\partial x\partial y},\frac{\partial^2\tilde{\alpha}}{\partial y\partial y}\right)$$



$\alpha_{Trained}$

$MSE(\alpha_{train},\tilde{\alpha})$

$\tilde{\alpha}\left(t_{i+1},\vec{x}_j\right)$