Progress Report

논문 지도교수: 윤군진 교수님

기계항공공학부 우주항공공학 전공

2014-17756 임상원

연구 주제와 이유

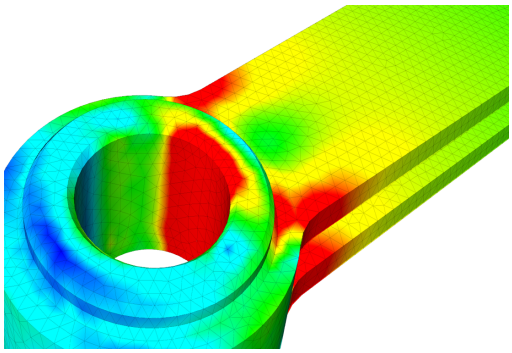
기공이 있는 재료의 미소 스케일 구조를 모델링하고, CNN을 이용해 image-base로 물성 예측

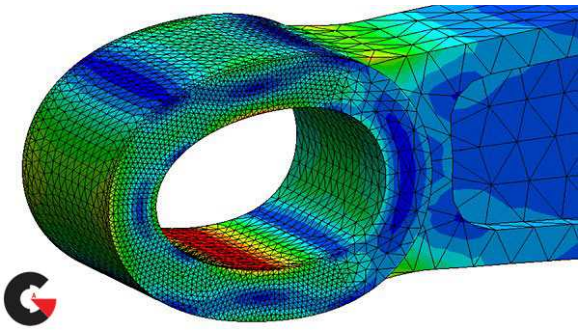
복잡한 구조의 composite material의 물성을 예측하는 것은 컴퓨팅 자원을 많이 소모하는 일이다. 기존의 접근 방식인 Finite Element Method에서는 물질을 일정한 크기의 최소 단위로 쪼개 이 단위들간의 상호작용을 연산하여 물성을 예측하는데, 미소 스케일부터 복잡한 composite 양상을 보이는 재료에 대해서는 지나치게 높은 연산비용이 들며, 높은 정확도를 보장하기 힘들다.

영상 분석 기술 발전의 일등 공신이라고 할 수 있는 Convolutional Neural Network 는 학습을 거치며 input 의 feature 를 자동으로 뽑아내고, 이를 바탕으로 분류 및 예측을 하는데 최적화되어 있다. 최근 많은 연구를 통해 3차원 필터를 활용하는 3-D CNN 을 활용하면 복잡한 구조물의 물성을 낮은 연산비용으로 예측할 수 있다는 점이 밝혀지고 있다.

따라서 이번 연구에서는 먼저 기공이 있는 재료의 미소 스케일 구조를 모델링하고, 3-D CNN 을 학습시켜 해당 재료의 물성을 예측하여 기존의 방식보다 낮은 컴퓨팅 자원과 높은 정확도를 얻어내고자 한다. 기존에 사용되는 정육면체의 filter 대신, 응력이 작용하는 축방향으로 길이가 긴 filter 를 활용해 축방향 응력에 대한 정보를 더 잘 예측할 수 있는지 알아보고, 여러 딥러닝 기법을 활용해 최적의 결과를 도출하고자 한다.

기존의 접근 방법 - FEA

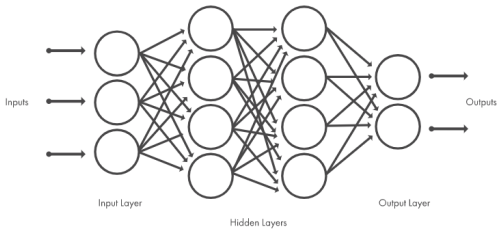




FEA(Finite Element Analysis) 는 공학분석에 사용되는 컴퓨터 시뮬레이션 기술이며 유한요소법이라고 불리는 수치적 기법을 사용한다. 복잡한 형상을 지닌 구조물의 물리적 특성을 예측하는데 주로 사용된다. 구조물이 복잡해질수록, 해석 영역 전체의 방정식을 한번에 세우는 것은 불가능에 가까워진다. 이 때문에 영역을 여러개의 단순하고 작은 격자(mesh)로 분할하고, 이 격자의 꼭짓점을 절점(node)으로 두어 각 격자들간의 상호작용을 계산한 것이 바로 FEA이다. 많이 사용되는 상용 소프트웨어로는 ABAQUS 가 있다.

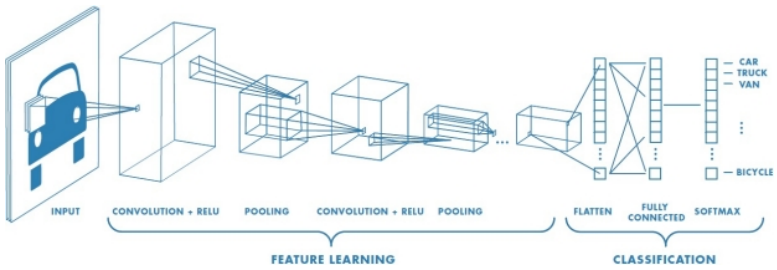
새로운 접근 방법 - CNN

CNN(Convolutional Neural Network) 란 기존의 FC(Fully Connected) 신경망의 단점을 보완하기 위해 나온 알고리즘이다.



<기존의 Fully Connected Layer>

FC 신경망은 데이터(주로 사진, 2차원 데이터)를 1차원 배열로 늘여놓은 뒤 학습을 진행하는데, 이로 인해 원래 중요한 정보들이 소실되고, 데이터가 조금만 바뀌어도 완전히 다른 데이터로 인식하는 등의 문제가 발생했다.

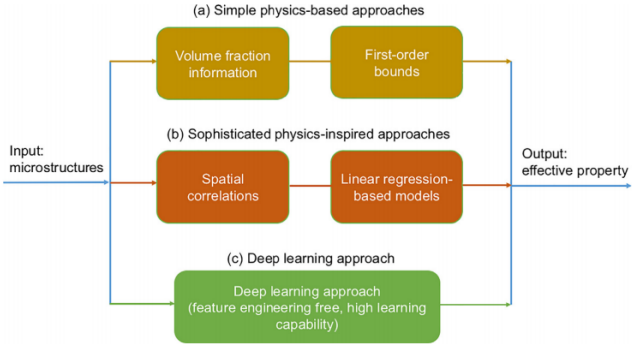


<CNN 의 구조도>

이를 해결하기 위해 나온 것이 바로 CNN 이다. CNN은 2차원 이미지의 모습을 그대로 유지한 채, 작은 크기의 filter를 이미지를 따라 이동시키며 연산을 진행한다. 학습은 바로 이 filter의 parameter 들을 바꾸는데, 각 filter 는 이미지에서 밝기, 가장자리, 직선, 곡선 등의 feature 를 추출하는 방향으로 학습된다.

$$y[n] = \sum_{a=-\infty}^{\infty} x[a]w[n-a]$$

<convolution 연산>



<기존의 연구 방법과 딥러닝을 이용한 방법의 모식도>

논문 리뷰

연구를 진행하기에 앞서 관련된 내용을 다루고 있는 다음 4개의 논문을 읽어보았다.

1. **Deep learning approaches for mining structure-property linkages in high contrast composites from simulation datasets**

   이 논문은 뛰어난 물리적 성질을 지닌 3차원 탄성 혼합 구조를 만들고자 하는 노력에서 출발한다. 이전 연구에서 feature-engineering based approach 를 한 데서 벗어나, CNN만을 활용하여 물성을 예측하고자 하였으며, 연구 결과 기존의 방법에 비해 크게 앞서는 결과를 보여준다

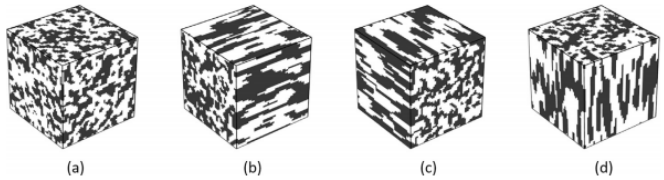


Fig. 1. Visualization of selected MVEs generated by applying different 3-D Gaussian filters to a random number field.

데이터셋은 binary 데이터로 생성하였으며, 각각 다른 성질을 지닌 두가지 재료로 구성된 구조를 의미한다. 개별 데이터셋의 크기는 연산비용의 부담을 지나치게 올리지 않는 선에서 최대한 크게 생성하여 미소 스케일에서 재료간의 상호작용을 충분히 볼 수 있도록 하였다. 위 사진에서 볼 수 있듯이 3개의 축에 대해 각기 다른 Gaussian filter를 적용해 서로 다른 방향성을 지닌 구조를 생성하였다.

각각의 데이터에 대한 Ground truth 는 Finite Element Model 을 적용한 결과를 사용하였다.

Activation Function 으로는 ReLU 를 사용했는데, 2개의 Conv layer 와 1개의 FC layer 로 구성된, 비교적 얕은 신경망에서 Gradient Descent 문제가 왜 발생했는지에 대한 의문이 들었다.

이외에도 L2 정규화를 이용해 overfitting 을 방지하였다.

결과를 보면, 3-D CNN 이 전통적인 방식(Simple physics-based approaches)에 비해 54% 앞선다는 점을 알 수 있다. (testing MASE) 3-D CNN 의 적용으로 높은 정확도, 낮은 컴퓨팅 비용, 높은 학습 유연성을 얻었다. 그러나 항상 최적의 결과만 나오는 것은 아니었는데, 이는 batch normalization, Residual structure, Inception structure 을 추후에 적용하여 개선할 수 있을 것이라 하였다. 또, Dropout 이 효과가 크게 없어서 DropConnect 나 stochastic pooling 을 적용해보고자 한다는 결론을 내린다.

2. **Establishing structure-property localization linkages for elastic deformation of three-dimensional high contrast composites using deep learning approaches**

이 논문에서도 역시 두가지 다른 물성을 지닌 재료를 혼합해 만든 여러 3-D 구조의 물성을 예측하는 연구를 진행했다. 앞선 연구와의 가장 큰 차이는 특정한 패턴이 없는 무작위의 구조를 생성하여 데이터로 사용한 점과, 두가지의 다른 contrast 를 지닌 데이터에 대해 각각 연구를 진행한 점, 그리고 기존의 머신러닝을 사용한 연구와의 정확도를 비교한 점이다.

데이터셋으로는 무작위로 생성된 3-D 구조를 사용하였으며, soft phase 는 0, hard phase 는 1 로 두어 행렬 형태로 만든 것으로 보인다. 각 구조에 대한 물성치, 즉 Ground Truth는 유한요소해석법을 이용해 구했다.

각각 다른 contrast 를 지닌 두 가지의 데이터셋(contrast-10, constrast-50)을 생성했다. contrast는 두 재료의 Young's Modulus 의 비를 의미하는데, contrast-10 이란 young's modulus 의 비율이 1:10 임을 의미한다.

CNN 모델의 구조는 다음과 같다(bs. = batch size)

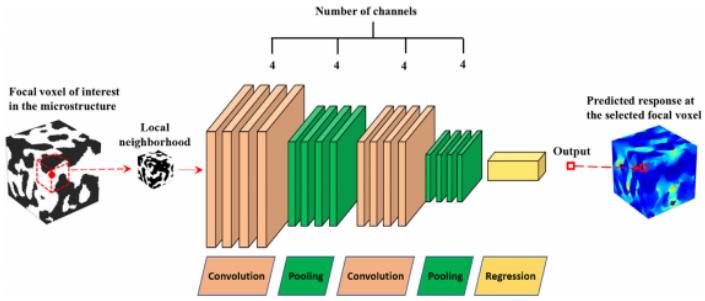


Fig. 2. Illustration of a typical convolutional neural network (CNN) used in this study for building localization linkages.

Table 1
The dimensionality of each layers in the proposed CNN (bs. Denotes the batch size).

Layer	Dimension
Input Layer	$bs. \times 11 \times 11 \times 11$
Convolutional Layer 1	$bs. \times 9 \times 9 \times 128$
Convolutional Layer 2	$bs. \times 7 \times 7 \times 256$
Fully Connected Layer 1	$bs. \times 2048$
Fully Connected Layer 2	$bs. \times 1024$
Output Layer	$bs. \times 1$

결과를 보면, contrast-10 과 contrast-50 데이터셋에 대한 MASE test 에서 기존의 머신러닝 방식을 각각 61.8%, 78.4% 앞서는 것을 알 수 있다.

MASE 란 mean absolute strain error 의 약자로 아래 식과 같이 정의된다.

$$e = \frac{1}{S}\sum_{s=1}^{S}\left|\frac{p_s - \hat{p}_s}{\langle \varepsilon_{11} \rangle}\right| \times 100\%$$

trained parameter 들은 transfer learning 에 응용될 수 있음(해당 모델의 데이터셋이 적다면 특히나 효율적)

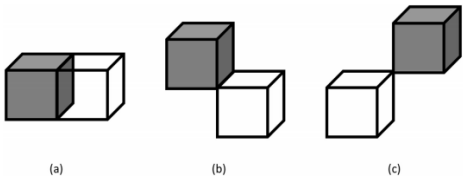
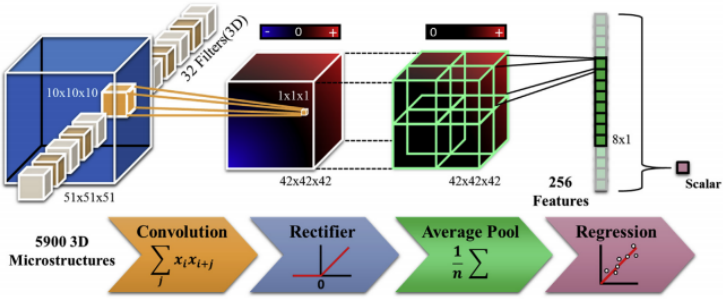


Fig. 8. Illustration of different level neighbors. (a) first, (b) second, and (c) third level neighbor.

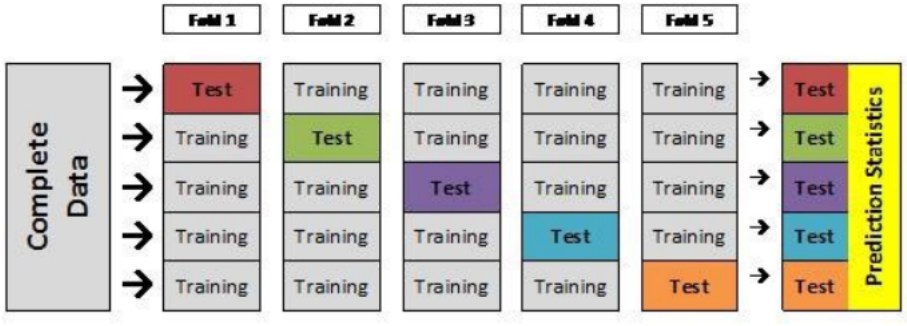
그러나 애초에 Ground Truth 가 FEA, 즉 시뮬레이션으로 예측한 결과이기 때문에 FEA에 비해 뛰어난지는 알 수 없다는 한계점을 제시한다. 정확한 결과를 위해서는 실제의 실험 데이터를 Ground Truth 로 사용해야 할 것이라고 저자는 주장한다. 그러나실험 데이터를 대량으로 얻는 것은 한계가 있으므로, 시뮬레이션으로 학습된 parameter 들을 transfer learning 에 활용한다면 적은 데이터셋으로도 좋은 결과를 낼 수 있지 않을까 하는 생각이 들었다.

3. **Material structure-property linkages using three-dimensional convolutional neural networks**

이 논문도 목적은 앞의 두 논문과 마찬가지로 뛰어난 물성을 지닌 구조를 만들기 위해 CNN으로 다양한 부재의 물성을 예측하는 것이다. 그러나 학습된 CNN을 이용해 higher order spatial statistics 의 정확도를 평가하고자 하는 시도와, CNN과 spatial correlations 를 복합적으로 활용해 신뢰도 높은 결과를 얻고, 목적을 달성하고자 한다는 점에서 차이가 있다.



3-D CNN 을 사용한 좋은 예시로, 모델에 대한 세세한 정보들이 잘 나와 있어 추후 연구에 참고하기 좋은 논문이었다. 다른 연구들과 마찬가지로 인공적으로 생성된 구조를 활용해 Finite Element 로 물성치를 구하고, 이를 CNN으로 구성된 인공신경망의 label 로 두어 학습을 진행했다.



<K-fold cross validation>

적은 데이터 수로 인한 제약을 극복하기 위해 10-fold cross validation 을 사용했다. 이는 전체 데이터를 10개로 나누어 각각의 test set들을 돌려가며 학습을 진행하는 방식이다. 데이터가 적은 환경에서 쉽게 발생하는 overfitting 을 방지하기 위한 노력으로 보인다.

2-point statistics 에서 추출하지 못하는 정보를 CNN 을 통해 추출할 수 있으며, 반대 역시 마찬가지로 성립하므로, 두가지 모델을 적절히 섞는 것이 더 나은 결과를 얻을 수 있음을 보여준다.

특이한 점은 1층의 Convolution layer 만을 사용했다는 점이다. 이미지의 복잡한 feature 를 추출하기 위해서 다층의 layer 를 쌓는 것이 일반적인데 아래 결과에서 나오다시피 filter가 단순히 원래 구조를 모방하는 방식으로만 학습이 일어난 것이 이런 이유에서가 아닐까 하는 의문이 들었다.

또, Average Pool 을 사용한 점도 눈에 띈다. 일반적으로 Average Pool 은 입력의 특징적인 부분을 주변과 평균냄으로서 강도를 약화시키므로, 다층의 Convolution layer 를 쌓을 때는 잘 쓰이지 않는데, 단층의 Convolution layer 라서 Average Pooling이 적용되어도 괜찮은 결과를 얻을 수 있지 않았나 하는 생각이 든다.

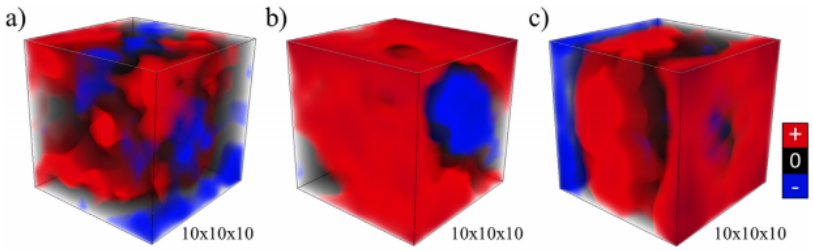


Fig. 8. Visualizations of a selected subset of learned filters that likely discriminated architectural features from microstructures similar to a), b) and c) in Fig. 2 respectively.

다른 논문과 다르게 이 논문에서는 학습된 각각의 CNN filter 를 시각화하여 분석했는데, filter의 구조가 데이터의 구조 중 일부의 모양을 닮는 방향으로 학습이 이뤄진다는 것을 알 수 있었으며, filter 중 일부는 불안정한 모습을 보이는데(fold에 따라 모습이 급변함, fig.8 (b)), 이는 데이터셋의 부족에 기인한 것으로 추측된다.

추후 연구에서 filter 를 시각화하여 hidden layer 속의 정보를 추출해야겠다는 아이디어를 얻을 수 있었다.

4. **Predicting the effective mechanical property of heterogeneous materials by image based modeling and deep learning**

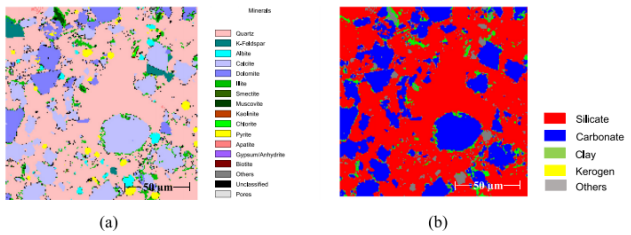


Fig. 3. (a) The SEM image of a mesoscale shale sample. (b) The corresponding simplified 5-phase model.

위 논문은 사진처럼 불균일한 내부 구조를 지닌 shale 의 물성을 CNN으로 예측하기 위한 연구이다. 평면 이미지, 즉 2-D 데이터에 대해 연구를 진행했으며, 다양한 물질을 함유하고 있는 shale을 위 그림처럼 5-phase model 로 간소화시켜 학습을 진행했다.

다른 연구와의 가장 큰 차이점은 500개의 실제 shale 샘플을 활용해 실험 결과를 검증하였다는 점이다. 0.97% 의 오차율을 보임으로서 CNN이 실제의 물성을 잘 예측함을 보여준 것이 큰 성과라고 할 수 있겠다.

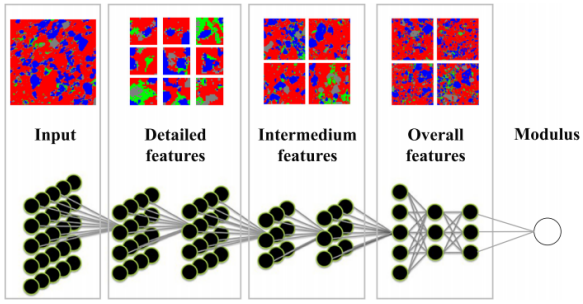


Fig. 11. A convolutional neural network to establish the implicit mapping between the mesoscale structure of a shale sample and its effective modulus.

그러나, 사용된 CNN 모델등에 대한 설명이 충분치 않다는 점이 아쉬웠다. 논문에 제시된 위 이미지로 미루어보아 다른 연구에 비해 다층의 CNN을 사용하였다고 추측할 뿐이다.

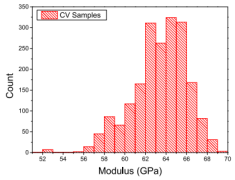


Fig. 13. The moduli distribution of the 2000 stochastic shale samples.

또, 실제 관측 결과의 Modulus 의 폭이 넓지 않다는 점을 짚고 넘어가고 싶었는데, 전혀 다른 구조와 물성을 지닌 shale 이 입력으로 들어왔을 때에도 적절한 물성을 예측할 수 있을지 의문이 들었다. 극단적인 경우를 대입했을 때에도 적절한 예측을 보여주거나, 아예 이상한 입력을 감지하는 요소를 추가한다면 더 나은 알고리즘이 되지 않을까 하는 생각이 들었다.

본 연구

먼저 본 연구에서는 2차원 기공 구조물의 물성을 예측하고 학습하는 CNN을 구성해보고, 제대로 작동함을 확인한 후 3차원 기공 구조물의 학습을 진행할 예정이다.

Neural Network 를 구성하기에 앞서 N % 의 Density 를 가지는 기공 구조물을 랜덤하게 생성하는 코드를 먼저 작성하였다. 1은 물질이 존재하는 부분, 0은 비어있는 부분으로 가정하였다.

```
# n percent dense 한 sample을 제작하는 코드
# size는 list
def NdenseArray(dim, size, density):
    if dim == 2:
        x_size = size[0]
        y_size = size[1]
        array = np.random.rand(x_size, y_size)
        for i in range(array.shape[0]):
            for j in range(array.shape[1]):
                array[i, j] = (1 if array[i, j] < density else 0)
        return array

    elif dim == 3:
        x_size = size[0]
        y_size = size[1]
        z_size = size[2]
        array = np.random.rand(x_size, y_size, z_size)
        for i in range(array.shape[0]):
            for j in range(array.shape[1]):
                for k in range(array.shape[2]):
                    array[i, j, k] = (1 if array[i, j, k] < density else
# example
NdenseArray(2, [10,10,10], 70
# output
"""
array([[1., 1., 0., 1., 0., 0., 0., 1., 1., 1.],
       [1., 1., 0., 1., 1., 1., 1., 1., 0., 1.],
       [1., 0., 1., 0., 1., 0., 1., 0., 1., 1.],
       [1., 1., 1., 1., 1., 1., 1., 0., 1., 0.],
       [0., 0., 1., 1., 1., 1., 0., 0., 1., 1.],
       [1., 1., 0., 1., 1., 1., 1., 0., 1., 1.],
       [1., 0., 1., 0., 1., 0., 0., 1., 0., 1.],
       [1., 1., 1., 1., 1., 1., 1., 1., 1., 1.],
```

```
       [0., 1., 1., 1., 0., 1., 0., 0., 1., 1.],
       [1., 1., 1., 0., 1., 1., 1., 1., 0., 1.]])
"""
```

생성된 array 의 실제 밀도를 측정하는 코드를 작성하였다.

```
# 생성된 array의 실제 밀도를 측정하는 코드
def density(array):
    dimension = len(array.shape)
    if dimension == 2:
        total = array.shape[0] * array.shape[1]
        count = 0
        for i in range(array.shape[0]):
            for j in range(array.shape[1]):
                if array[i, j] == 1:
                    count += 1
    elif dimension == 3:
        total == array.shape[0] * array.shape[1] * array.shape[2]
        count = 0
        for i in range(array.shape[0]):
            for j in range(array.shape[1]):
                for k in range(array.shape[2]):
                    if array[i, j, k] == 1:
                        count += 1
    return count/total*100
```

이들을 통해 특정 밀도로 채워진 기공 구조물을 행렬로 모델링할 수 있다. 현재는 단순히 랜덤하게 생성하므로 모델의 크기가 커진다면 비슷한 내부 구조로 수렴할 것이라 추측할 수 있다.

이 때문에 다른 연구에서 제시되었던 것처럼 축방향에 따라 다른 확률분포를 가지는 구조를 생성하는 코드를 추가로 작성해나갈 예정이다.

CNN 을 이용한 2차원 기공 구조물의 물성 예측

우선 2차원 기공 구조물을 먼저 학습시켜보고, 3차원 기공 구조물에 대한 학습을 진행하려고 한다.

Keras 라는 Deep learning API 를 사용해 손쉽게 신경망을 구성하였다. 이는 간단하게 신경망을 구성할 수 있다는 장점이 있으나, 신경망을 세세하게 조정하기에는 어려움이 따를 수 있다는 생각이 들었다. 이 때문에 따로 CNN 을 python 과 numpy 모듈만을 활용해 기초부터 구성해나가려고 한다. 현재 진행 상황은 아래에 코드로 첨부하겠다.

위의 코드에서 생성된 기공 구조물 데이터들을 train_images, test_images 로 활용하고, FEA를 통해 얻은 구조의 물성치들을 train_labels, test_labels 로 활용하여 학습을 진행할 예정이다.

label 은 ABAQUS 를 사용해 구할 예정이며, 추후에 해당 툴의 사용법을 익히고 내용 보충해 나가도록 하겠다.

```
# Simple keras CNN for Analyzing Pore structure
import tensorflow as tf
from tensorflow.keras import datasets, layers, models

# input (2차원 이미지)
# 우선 랜덤한 값을 넣어뒀는데, 위에서 만든

# 랜덤하게 만든 이미지의 크기
x_input = 50
y_input = 50

# 랜덤으로 생성된 2-D 이미지, 1은 차있는 부분, 0은 비어있는 부분을 의미
train_images = tf.random.normal((10000, x_input, y_input, 1))
# 해당 이미지의 물성치, Finite Element 로 계산
train_labels = tf.random.normal((10000,))

# 테스트 이미지도 마찬가지로 생성한다
test_images = tf.random.normal((100, x_input, y_input, 1))
test_labels = tf.random.normal((100,))


# 모델 구성
model = models.Sequential()
# Convolution layer - padding 이 없으므로 x, y 방향 크기가 각각 2씩 줄어든다
model.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=train_images.shape[1:]))
# 2x2 Maxpooling layer - 이미지의 크기가 절반으로 준다
model.add(layers.MaxPooling2D((2,2)))
# Convolution layer - padding 이 없으므로 x, y 방향 크기가 각각 2씩 줄어든다
```

```
model.add(layers.Conv2D(64, (3,3), activation='relu'))
# 2x2 Maxpooling layer - 이미지의 크기가 절반으로 준다
model.add(layers.MaxPooling2D((2,2)))
# Convolution layer - padding 이 없으므로 x, y 방향 크기가 각각 2씩 줄어든다
model.add(layers.Conv2D(64, (3,3), activation='relu'))
# 2x2 Maxpooling layer - 이미지의 크기가 절반으로 준다
model.add(layers.MaxPooling2D((2,2)))
# 2차원 이미지를 1차원으로
model.add(layers.Flatten())
# Fully Connected Layer
model.add(layers.Dense(64, activation='relu'))
# Fully Connected Layer
model.add(layers.Dense(10, activation='softmax'))

model.summary()

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy', # Cross Entropy 를 사용해 더 빠른 학습속도를 기대.
              metrics=['accuracy'])

history = model.fit(train_images, train_labels, epochs=100)
# 추후에 구현할 것: test_acc 가 올라가기 시작하면 overfitting 이 일어나고 있다는 것이므로 학습 중단
# parameter 의 시각화

test_loss, test_acc = model.evaluate(test_images,  test_labels, batch_size=32, verbose=1)
```

위에서 말한 것 처럼, Keras API 만 사용해서는 해결하지 못하는 문제가 생길 수 있어, 따로 CNN 에 대한 학습 차원에서 CNN에 적용되는 개념인 convolution, padding, pooling 등을 구현해 보았다.

```
# max-pooling
def max_pooling(array, filter_size=2, stride=2):
    dimension = len(array.shape)
    # input 은 짝수개로 들어오도록 하자
    if dimension == 2:
        result = np.zeros((int(array.shape[0]/stride), int(array.shape[1]/stride)))
        for i in range(0, array.shape[0], stride):
            for j in range(0, array.shape[1], stride):
                result[int(i/stride), int(j/stride)] = np.max(array[i:i+filter_size, j:j+filter_size])
    if dimension == 3:
        result = np.zeros((int(array.shape[0]/stride), int(array.shape[1]/stride), int(array.shape[2]/stride)))
        for i in range(0, array.shape[0], stride):
            for j in range(0, array.shape[1], stride):
                for k in range(0, array.shape[2], stride):
                    result[int(i/stride), int(j/stride), int(k/stride)] = np.max(array[i:i+filter_size, j:j+filter_size, k:k+fi
    return result
```

```
# zero-padding (2D, 3D)
def zero_padding(array, padding_size=1):
    dimension = len(array.shape) # dimension
    print("dimension:", dimension)
    if dimension == 2:
        result = np.zeros((array.shape[0] + padding_size*2, array.shape[1] + padding_size*2))
        result[padding_size: padding_size + array.shape[0], padding_size: padding_size+array.shape[1]] = array
        return result

    elif dimension == 3:
        result = np.zeros((array.shape[0] + padding_size*2, array.shape[1] + padding_size*2, array.shape[2] + padding*2))
        result[padding_size: padding_size + array.shape[0], padding_size: padding_size+array.shape[1], padding_size: padding_si
        return result
```

3D CNN 을 이용한 3차원 기공 구조물의 물성 예측

작성 예정

- filter 의 모양을 어떻게 해야 더 좋은 결과가 나올지 시도해보고자 한다.
- 축방향 강도가 중요하다면 cnn filter 의 차원을 (3×3x축방향 길이) 로 하는게 좋지 않을까?

기존 방식과의 비교, 나아진 점

작성 예정

향후 연구 방향, 개선 사항

작성 예정

- 학습시킨 모델을 데이터셋이 부족한 다른 연구 사례에 적용시켜 transfer learning 을 시도해본다

Final Report 제출 전 계획

- 연구의 학술적 기반을 수식으로 정리한다(Convolution, Cross Entropy, Kullback Leibler Divergence, Finite Element Analysis 등)
- 재료의 3-D 구조를 보다 복잡하게 구성한다. 방향에 따른 불균일성 등을 생성 코드에 추가한다.
- ABAQUS 를 활용해 Label 로 사용될 데이터를 구한다.
- 3-D CNN 을 학습시킨다.
- hyperparameter 들을 조정하며 fine-tuning 을 이어나간다.
- (연구 장비를 이용할 수 있다면) 실제 재료들을 가지고 실험을 진행한 후, 학습된 모델의 결과와 비교해본다.

일정

Aa 일정	계획
10/21 - 11월 초	ABAQUS 사용법 학습
11월 초 - 11월 중	3D CNN 학습, 재료 구조 복잡화
11월 중 - 11월 말	fine-tuning, 연구 내용 정리
11월 말- 12/4	Final Report 보완 및 작성