초 록

이 연구는 추가정의 최적 시추 위치를 선정하는 과정에서 저류층 시뮬레이션 기반의 딥러닝 프록시 기법을 제안한다. 전형적인 다층구조 인공신경망은 입력값의 다차원 공간 분포를 보전하지 못하고 1차원 배열로 변환하여 학습하는 한계가 있었다. 반면, 합성곱 신경망은 다차원 배열 형태의 입력값에서 공간 분포의 특징을 추출하는데 탁월하다. 합성곱 신경망을 사용하여 유정 인근의 물성 분포와 유정의 생산성 사이의 상관관계를 학습하고, 학습된 신경망 모델에 생산성이 학습되지 않은 좌표 주위의 물성 분포를 입력하면 해당 좌표에 추가정이 시추된 경우의 생산성을 예측할 수 있다. 이 연구에서는 정적 물성(예: 유체투과도)과 동적 물성(예: 오일 포화도)을 통합적으로 고려하기 위하여 다중모달 학습을 합성곱 신경망에 적용하여 다양한 종류의 입력값을 분석할 수 있는 신경망을 설계하였다.

제안한 기법을 SPE10 벤치마크 저류층에 적용하여 추가정의 최적 위치 선정을 수행하였다. SPE10 상부의 근해 부분과 하부의 채널 부분을 분리하여 개별 저류층 모델로 구성한 후 각각 학습을 진행하였다. 제안한 다중모달 합성곱 신경망의 시험 성능은 근해와 채널 저류층 모두에서 우수하였으며, 특히 채널 저류층에서 인공신경망에 비해 우수한 성능을 보였다. 이것은 채널 저류층에서 나타나는 불연속적인 채널의 경계면이 영상 내 사물 인식 분야의 사물의 윤곽선과 유사한 역할을 하기 때문으로 해석하였다. 학습된 합성곱

신경망 모델을 사용하여 도출한 생산 유망 지점에 대해 저류층 시뮬레이션을 수행하여 검증하였다. 근해 저류층과 채널 저류층 각각에서 합성곱 신경망이 도출한 상위 20개 지점에 대해 저류층 시뮬레이션을 수행하여 검증한 결과, 근해 저류층에서의 평균 오차는 1.50%였으며 채널 저류층에서의 평균 오차는 이상 현상을 제외할 시 4.27%였다. 이러한 검증 결과를 기반으로 생산성이 가장 높은 지점을 선택함으로써 최적의 추가정 시추 위치를 도출할 수 있다.

이 연구에서는 자료기반 기계 학습 기법인 합성곱 신경망을 사용해 유정 인근 물성의 공간적 분포와 유정의 생산성 사이의 관계를 학습하였다. 또한, 복잡한 비선형적 관계를 모사하기 위하여 다중모달 학습을 적용하여 정적 물성과 동적 물성을 동시에 고려하였다. 이를 통하여 합성곱 신경망뿐 아니라 인공신경망도 기존의 연구 결과보다 훨씬 향상된 성능을 보였다. 상기한 점에서 기존의 연구와 차별성을 가지는 본 연구는 다양한 종류의 저류층 시뮬레이션 기반 문제에 적용되어 효과적이고 신속한 의사결정 수단으로 활용될 수 있을 것으로 기대된다.

주요어: 추가정 위치 최적화, 딥러닝 프록시, 합성곱 신경망, 다중모달 학습

학번: 2011-21115

목 차

List of Tables

List of Figures

1. 서론

추가정(infill well)의 시추는 오일 생산량 또는 순현재가치(net present value)를 최대화하기 위해 석유개발 현장에서 빈번하게 이루어진다. 추가정의 최적 위치 선정을 포함한 석유개발 과정에서의 여러 종류의 의사결정에 있어서 저류층 시뮬레이션이 널리 사용된다. 저류층 시뮬레이션은 주요 물리 법칙과 여러 입력 자료로부터 저류층의 거동을 예측하기 때문에 다양한 석유공학 연구에서 주요한 수단으로 사용하고 있다. Alusta 등(2011)은 노후 저류층(depleted reservoir)에서 폴리머 공법(polymer flooding)과 추가정 시추의 경제적 효과를 비교하였다. Mahmoud(2015)는 유체투과도와 공극률, 물 포화도를 결합한 초물성(super-property)을 제안하여 정적 물성의 불확실성을 줄임으로써 추가정의 생산량 예측 성능을 높였다. Safari 등(2017)은 치밀 저류층에서 응력장의 변화에 기초하여 추가정의 시추 스케줄 및 파쇄대의 디자인을 최적화 하였다. 그러나 저류층 시뮬레이션에 기반한 의사 결정 방법은 많은 수의 시뮬레이션 수행을 필요로 하기 때문에 신속한 의사 결정이 어렵다는 단점이 있다. 이러한 한계를 보완하기 위해 회귀 분석(regression), 크리깅(kriging), 인공신경망(artificial neural network, ANN)과 같은 다양한 종류의 프록시 기법이 저류층 시뮬레이션을 대체하기 위해 사용되어 왔다(Maysami et al., 2013; Korjani et al., 2016; Polizel et al., 2017).

인공신경망은 1990년대 이후 여러 분야에서 널리 사용된 기계 학습(machine learning) 기법으로 범용성이 매우 뛰어나기 때문에 석유공학에서도 다양한

문제에 적용되었다. Luthi와 Bryant(1997)는 유정 검층 자료를 사용해 지질학적 표식(geological marker)의 위치를 예측하였다. Al-Fattah와 Startzman(2001)은 생산정의 수, 탐사정의 수, 유가, 천연가스 생산감퇴율, 매장량 등을 바탕으로 20년 후의 천연가스 생산량을 예측하였다. 한편, PVT 자료 사이의 관계를 매칭하는 연구도 꾸준히 진행되었다(Gharbi and Elsharkawy, 1997; Osman et al., 2001; Asadisaghandi and Tahmasebi, 2011). Xue 등(2014)은 유전 알고리듬과 인공신경망을 결합하여 유정 검층 자료로부터 균열의 물성을 예측하였고, Wang과 Chen(2016)은 유정 완결 정보로부터 수압파쇄 유정의 성능을 예측하였다. Eskandarian 등(2017)은 랜덤 포레스트(random forest)와 인공신경망을 사용하여 굴진율(rate of penetration)을 평가하였다. 유정의 위치 최적화에 인공신경망을 사용한 사례는 다음과 같다. Centilmen 등(1999)은 다중 유정의 저류층 시뮬레이션 결과를 인공신경망으로 학습하여 최대의 가스 생산량을 산출하였다. Yeten 등(2003)은 유전 알고리듬과 인공신경망을 결합하여 비전통 유전에서의 유정의 종류와 위치, 경로를 최적화하였다. Min 등(2011)은 퀄리티 맵(quality map)을 인공신경망의 입력값으로 사용하여 다중 추가정의 최적 시추 위치를 도출하였다. Jang 등(2018)은 수평정의 최적 위치 선정을 위하여 순차적 학습(sequential training)을 인공신경망에 접목하였다. 이상의 전형적인 다층구조 인공신경망을 사용한 연구들은 모든 입력값을 하나의 1차원 배열 형태로 변형하여 사용하여 왔다. 개별 값이 각각의 현상을 나타내는 경우에는 하나의 배열로 만들어 입력하더라도 문제가 발생하지 않지만, 유정 인근의 유체투과도

분포와 같이 다수의 값이 하나의 현상을 나타내는 경우에는 1차원 배열의 형태로는 입력값의 공간적 분포 특징을 적절히 반영할 수 없는 문제가 발생한다. 이러한 이유로 인하여 인공신경망은 높은 범용성에도 불구하고, 실질적으로는 제한된 종류의 입력 자료만을 사용하는 한계를 가진다. 이 외에도 일반적인 인공신경망에서는 기울기 소실(vanishing gradient)과 과적합(overfitting)과 같은 문제가 발생하여 왔다(Hochreiter, 1998; Hawkins, 2004).

최근 딥러닝(deep learning)은 기존의 인공신경망의 한계를 극복하고 사물 인식이나 음성 인식 분야 등에서 높은 성능을 보이고 있다(Glorot et al., 2011; Hinton et al., 2012). 합성곱 신경망(convolutional neural network, CNN)은 가중치 공유를 통해 입력값의 특징을 효율적으로 추출하는 대표적인 딥러닝 기법으로 이미지 처리나 사물 인식 분야에서 매우 뛰어난 성능을 보인다(LeCun et al., 1998). 기존의 인공신경망에 비해 합성곱 신경망이 가지는 가장 큰 특징은 입력값으로 1차원 배열이 아닌 다차원 배열을 사용하는 것이다(Behnke, 2003). 합성곱 신경망은 컨볼루션(convolution)과 풀링(pooling)이라는 과정을 통해 입력된 배열에서 특징을 추출하고 압축한다. 이를 통해 합성곱 신경망은 입력값 사이의 패턴이나 신호를 찾아낼 수 있다. 지도 학습의 일종인 합성곱 신경망은 학습 자료의 출력값 형태에 따라 분류(classification)와 회귀(regression) 모형으로 나눌 수 있다(Figueiredo, 2003). 분류 모형의 경우 출력값은 분절된 값으로 나타나며, 일반적으로 입력값이 속하는 클래스를 나타내는 인덱스의 형태를 띈다. 반면, 회귀 모형에서는 특정 유형의 연속적인 값이 출력값으로 사용된다.

비교적 새로운 기법인 합성곱 신경망을 석유공학에 적용한 사례는 많지 않다. Hass와 Hustad(2018)는 16,000여 장의 생산 설비(너트, 용접, 파이프 등) 영상을 학습한 합성곱 신경망을 사용해 외관 검사(visual inspection)를 수행하였다. Revelas 등(2018)은 해저(sea floor)의 영상으로부터 퇴적물의 프로필과 입자의 크기를 분류하였다. Pennel 등(2018)은 시계열 자료를 사용해 인공채유(artificial lift)의 상태를 진단하고 예측하였고, Sundararaman 등(2018)은 라이저(riser)의 변형율과 움직임을 모니터링하고 예후를 통합적으로 관리하였다. 앞서의 연구들은 모두 분류 모형의 합성곱 신경망을 사용하였으며, 오직 한 종류의 입력값을 사용하였다. 일반적인 합성곱 신경망의 구조는 동시에 한 종류의 입력값만을 처리할 수 있지만, 추가정의 생산량 예측에는 하나 이상의 요소가 영향을 미친다. 따라서 합성곱 신경망이 다양한 종류의 입력값을 동시에 처리할 수 있도록 다중모달(multi-modal) 학습을 적용할 필요가 있다.

인공지능 분야에서의 다중모달 학습이란 다양한 양식의 자료를 통합하여 모델링하는 것을 의미한다(Srivastava and Salakhutdinov, 2012). Ngiam 등(2011)은 비디오 자료(예: 사람의 발음 신호)와 오디오 자료(예: 발음할 때의 입술 영상)를 통합하여 음성 인식 성능을 향상시켰다. Srivastava와 Salakhutdinov(2012)는 심층 볼츠만 머신(deep Boltzman machine)을 사용하여 이미지와 문장을 결합함으로써 정보 전달력을 향상시켰다. Sohn 등(2014)은 기존 자료 양식으로부터 누락된 자료 양식을 예측하는 자료 생성 모델을 제안하였다. Ma 등(2015)은 합성곱 신경망을 사용하여 이미지와 문장을 결합하였다. Eitel 등(2015)은 컬러(RGB)

영상과 깊이(depth) 영상을 결합하여 사물 인식 성능을 높였다. Zhu 등(2017)은 사람의 제스쳐를 감지하기 위하여 합성곱 신경망과 LSTM(long short term memory) 네트워크를 결합하였다. 이러한 연구에 기초하여, 정적 자료(유체투과도, 공극률 등)와 동적 자료(압력, 포화도 등)와 같이 양식이 서로 다른 자료를 동시에 합성곱 신경망에 입력하고 통합적으로 분석한다면 높은 예측 성능을 보일 것으로 기대된다.

이 연구의 주요 목적은 추가정의 최적 위치를 신속하게 선정하기 위해 딥러닝 기법을 사용해 저류층 물성의 분포로부터 생산성을 최대화할 수 있는 추가정의 위치를 탐색함으로써 저류층 시뮬레이션 횟수를 최소화하는데 있다. 이를 위해 공간적 물성 분포를 분석할 수 있는 합성곱 신경망 기법을 채택하였고, 동시에 다양한 물성 자료를 통합적으로 분석하기 위하여 다중모달 학습을 적용하였다. 더욱 구체적으로 추가정 인근의 정적 및 동적 물성 분포와 유정의 위치에 기반한 거리 정보를 입력값으로, 유정의 생산량을 출력값으로 가지는 학습 자료를 사용해 다중모달 학습이 적용된 합성곱 신경망을 학습한 후 학습되지 않은 지점에 대해 생산량을 예측함으로써, 추가정의 최적 위치 선정에 필요한 저류층 시뮬레이션 횟수를 줄일 수 있다.

이 논문은 총 5장으로 구성된다. 제 1장 서론에서는 이 연구의 필요성과 목적, 기존 연구들의 한계에 대하여 서술하였다. 제 2장에서는 기계 학습과 인공신경망, 합성곱 신경망의 이론적 특징을 설명한다. 제 3장에서는 이 연구에서 채택한 다중모달 학습이 적용된 합성곱 신경망을 사용하여 추가정의

생산량 예측에 대하여 서술한다. 제 4장에서는 연구 결과를 나타내었다. 마지막으로 제 5장 결론에서는 연구 결과로부터 결론을 도출한다.

2. 이론적 배경

본 장에서는 자료기반 기계 학습과 인공신경망, 합성곱 신경망에 대해 설명한다. 학습 자료를 통해 기계가 스스로 학습하는 기계 학습 분야의 대표적인 기법이 인공신경망이며, 최근의 딥러닝은 인공신경망의 일부분이다. 대표적인 딥러닝 기법으로 합성곱 신경망, 순환 신경망(recurrent neural network), 오토인코더(autoencoder) 등이 있으며, 이 논문에서는 합성곱 신경망을 사용해 유정의 생산성을 평가하는 연구를 다룬다.

2.1. 자료기반 기계 학습

자료기반 기계 학습은 명시적 프로그래밍 없이 기계, 즉 컴퓨터가 학습 자료로부터 스스로 모델이나 법칙을 도출하는 것을 의미한다(Samuel, 1959). Figure 2.1은 기계 학습의 세 종류 학습 방식인 지도 학습(supervised learning), 비지도 학습(unsupervised learning), 강화 학습(reinforcement learning)을 나타낸 그림이다. 지도 학습은 기계 학습 분야에서 가장 먼저 연구된 분야로, 학습 자료는 입력값과 이에 대응하는 출력값(참값)의 쌍으로 구성된다. 기계 학습 모델은 입력값을 입력 받으면 출력값(예측값)을 도출한다. 해당 입력값에 대해서 모델이 도출해야 하는 출력값(참값)이 학습 자료로 주어지기 때문에 예측값과 참값의 차이를 비교할 수 있다. 이러한 차이를 손실 함수(loss function) 또는 비용 함수(cost function)라고 불리는 목적 함수 형태로 정량화한 후, 손실

함수값을 최소화하는 방향으로 모델을 수정하는 과정을 반복함으로써 기계 학습 알고리듬이 학습된다. 이러한 지도 학습 방식을 채택하는 알고리듬들의 활용은 분류(classification)나 회귀 분석(regression)이 대표적이다. 해당 학습 방식을 사용하는 기법으로는 합성곱 신경망이 대표적이다. 비지도 학습의 학습 자료는 입력값만이 존재하고 출력값이 존재하지 않는다. 따라서 주로 자료의 특성을 분석 및 가공하여 자체적인 기준을 만들기 위해 사용한다. 비지도 학습은 일반적으로 군집화(clustering)나 차원 축소에 사용한다. 또한, 불량 제품을 선별하는 이상 탐지(anomaly detection)에도 사용한다. 대표적인 기법으로 k-means clustering이나 오토인코더가 있다. 강화 학습은 학습 자료가 입력값과 출력값, 출력값에 대한 평가로 구성되어 있다. 평가를 통해 보상을 제공함으로써 주로 자세 제어나 게임 플레이 등의 상호작용에서의 최적의 선택을 학습하는데 사용한다. 강화 학습의 대표적인 기법으로는 DQN(Deep Q Network)이 있다.

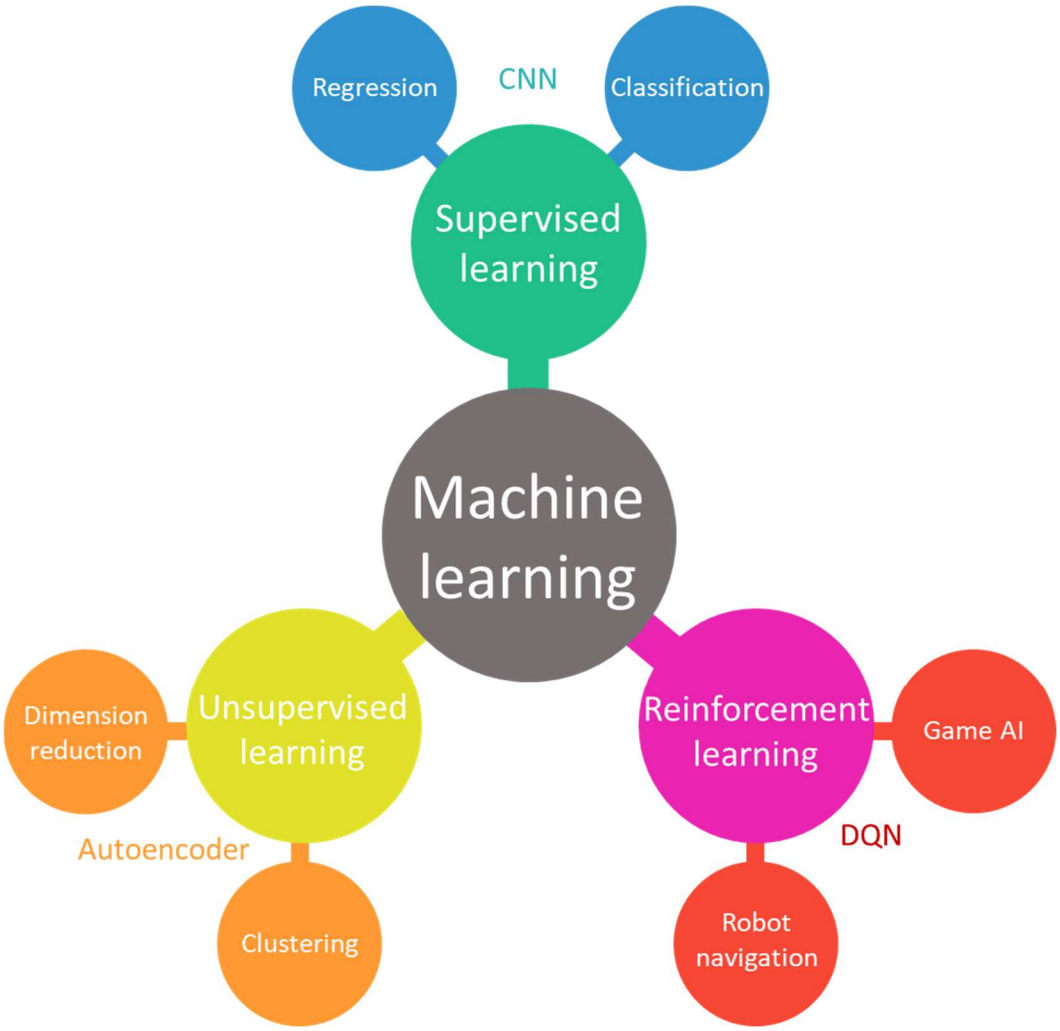


Figure 2.1 A variety of learning approaches in machine learning.

Figure 2.2는 기계 학습과 인공신경망, 딥러닝 사이의 관계를 나타낸다. 기계 학습의 대표적인 분야로 인공신경망이 수십 년 동안 연구되었으며, 최근 각광받고 있는 딥러닝은 인공신경망의 한 분야를 일컫는다. 딥러닝의 기본 원리는 기존의 인공신경망과 크게 다르지 않지만, 다층(multi-layer) 구조를 차용하여 심층적 분석을 수행함으로써 사물 인식과 음성 인식 분야 등에서 탁월한 성능을 보이고 있다. 이 연구는 지도 학습 기반의 인공신경망, 그 중에서도 딥러닝 기법에 초점을 둔다.

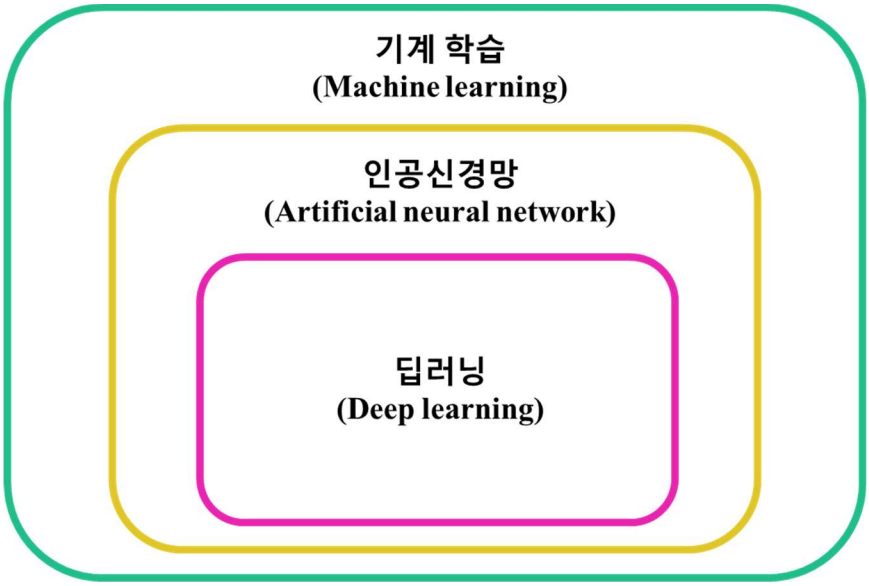


Figure 2.2 Relationship between machine learning, artificial neural network, and deep learning.

2.2. 인공신경망

인공신경망은 기계 학습의 한 분야로서 인간의 신경세포 연결 형태에서 착안되었다(Rosenblatt, 1958). Figure 2.3은 생물학적 신경세포와 인공신경망의 신경망 구조를 나타낸다. 인공신경망은 생물학적 신경세포의 신호 전달 과정을 모방하여 비선형적 관계를 가진 입력 정보와 출력 정보를 처리한다. 인공신경망의 신경세포는 노드(node)라고 불리며, 정보 전달 체계의 기본 요소이다. 각 노드는 입력값 x와 가중치 W 의 곱의 선형합을 통해 출력값 y를 도출한다. σ 는 활성화 함수(activation function)로 입력값과 출력값 사이에 비선형적 관계를 부여하는 역할을 한다. 활성화 함수에 대한 상세한 설명은 후술한다.

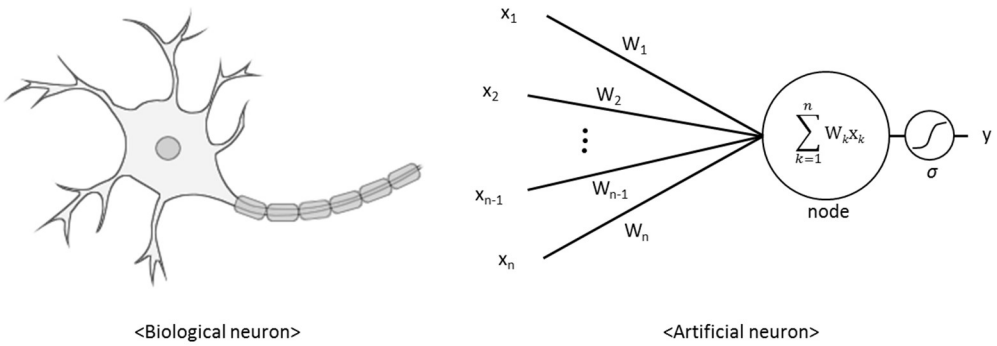


Figure 2.3 Comparison of biological and artificial neurons.

Figure 2.4는 다수의 노드가 모여서 이루어진 다층 인공신경망(multi-layer ANN)의 전형적인 구조이다. 인공신경망은 한 개의 입력층(input layer)과 한 개 이상의 은닉층(hidden layer), 한 개의 출력층(output layer)으로 구성된다. 입력층과 출력층은 전체 신경망의 입출력을 담당하고, 은닉층은 실질적인 학습과 예측 작업을 수행한다. 은닉층은 해당 층의 출력값이 직접적으로 드러나지 않기 때문에 붙여진 명칭이다. 입력층에서 출력층으로의 방향으로 인접한 층 사이에서 연산이 순차적으로 진행되며, 각 층의 출력값은 다음 층의 입력값으로 사용된다. 각각의 층은 다수의 노드로 구성되며, 각 층의 노드는 인접한 층의 모든 노드와 가중치(weight)로 연결된다. 각각의 노드에서는 하나씩의 연산이 수행되며, 가중치는 노드 간의 연결 강도를 의미한다.

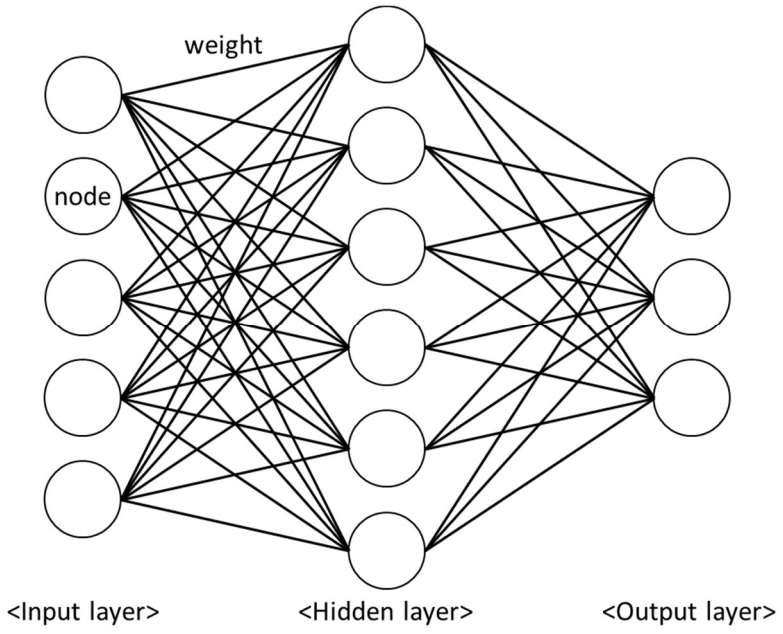


Figure 2.4 Structure of ANN.

인공신경망의 학습은 손실 함수를 최소화하는 비선형적 관계를 찾아내기 위하여 가중치를 조절하는 과정을 뜻한다(Min et al., 2011). 손실 함수를 사용해 모델을 학습하는 알고리듬으로는 주로 역전파(back-propagation) 기법을 사용한다(LeCun et al., 1989). 역전파 기법은 모델의 출력값과 참값 사이의 오차를 계산하는 과정인 전향 단계와 경사 하강법(gradient descent)을 사용해 오차가 작아지도록 가중치를 갱신하는 과정인 후향 단계를 반복하여 학습을 진행한다. 경사 하강법은 함수를 미분하여 기울기를 구하고 기울기가 낮은 방향으로 조금씩 인자를 바꾸면서 최적해를 찾는 방법이다. 이때 이전 학습에서의 영향을 반영하는 정도를 학습률(learning rate)이라고 한다. 역전파 기법은 원칙적으로 최종 출력층에서만 정의되는 오차를 델타 규칙(Delta rule)에 따라 은닉층에서도 정의함으로써, 모든 층에서 학습을 진행한다(Stone, 1986).

인공신경망의 임의의 층의 전향 단계의 연산 과정은 식 (1)과 같이 나타낼 수 있다.

$$\mathrm{Y} = \sigma(\mathrm{WX} + \mathrm{B}), \tag{1}$$

이 때 Y는 해당 층의 1차원의 출력값 배열이며, σ는 활성화 함수, W는 노드 사이의 2차원의 가중치 배열이다. X는 연산이 이루어지는 층으로의 1차원의 입력값 배열이며, B는 활성화 함수의 출력값을 조절하는 1차원의 편향(bias) 배열이다. 만일 $N_1 \times N_2 \times N_3$와 같은 다차원 배열이 입력값으로 사용된다면, 해당 배열은 $N_t \times 1$의 1차원 배열로 변환되어 입력하게 된다($N_t = N_1 \times N_2 \times N_3$).

인공신경망의 활성화 함수의 역할은 입력값과 출력값 사이의 복잡한 관계를 모사하기 위하여 비선형성을 추가하는 것이다(Specht, 1990). Figure 2.5는 인공신경망에서 주로 사용되는 네 종류의 활성화 함수, sigmoid, hyperbolic tangent(tanh), linear, rectified linear unit(ReLU)을 나타낸다. Sigmoid 함수와 tanh 함수는 출력값의 최댓값이 1인 S자 형태의 함수로서 기존의 인공신경망에서 주로 사용되었다(Mass et al., 2013). 최근에는 기울기 소실(vanishing gradient) 문제를 해결하기 위해 많은 딥러닝 기법에서 ReLU를 사용하고 있다(Glorot et al., 2011). ReLU 함수는 양수는 그대로 출력하고 음수에 대해서는 0을 출력한다. 기존의 sigmoid 함수나 tanh 함수의 경우 출력값에 제한이 있고 강한 출력 부분에서의 미분값이 0에 가까웠기 때문에 은닉층의 수가 늘어날수록 강한 출력이 다음 층으로 잘 전달되지 않는 기울기 소실 문제가 발생하였다. 반면에 ReLU 함수는 출력에 제한이 없기 때문에 다수의 층으로 구성된 심층신경망 구조에서 기울기 소실 문제를 효과적으로 해결하였다.

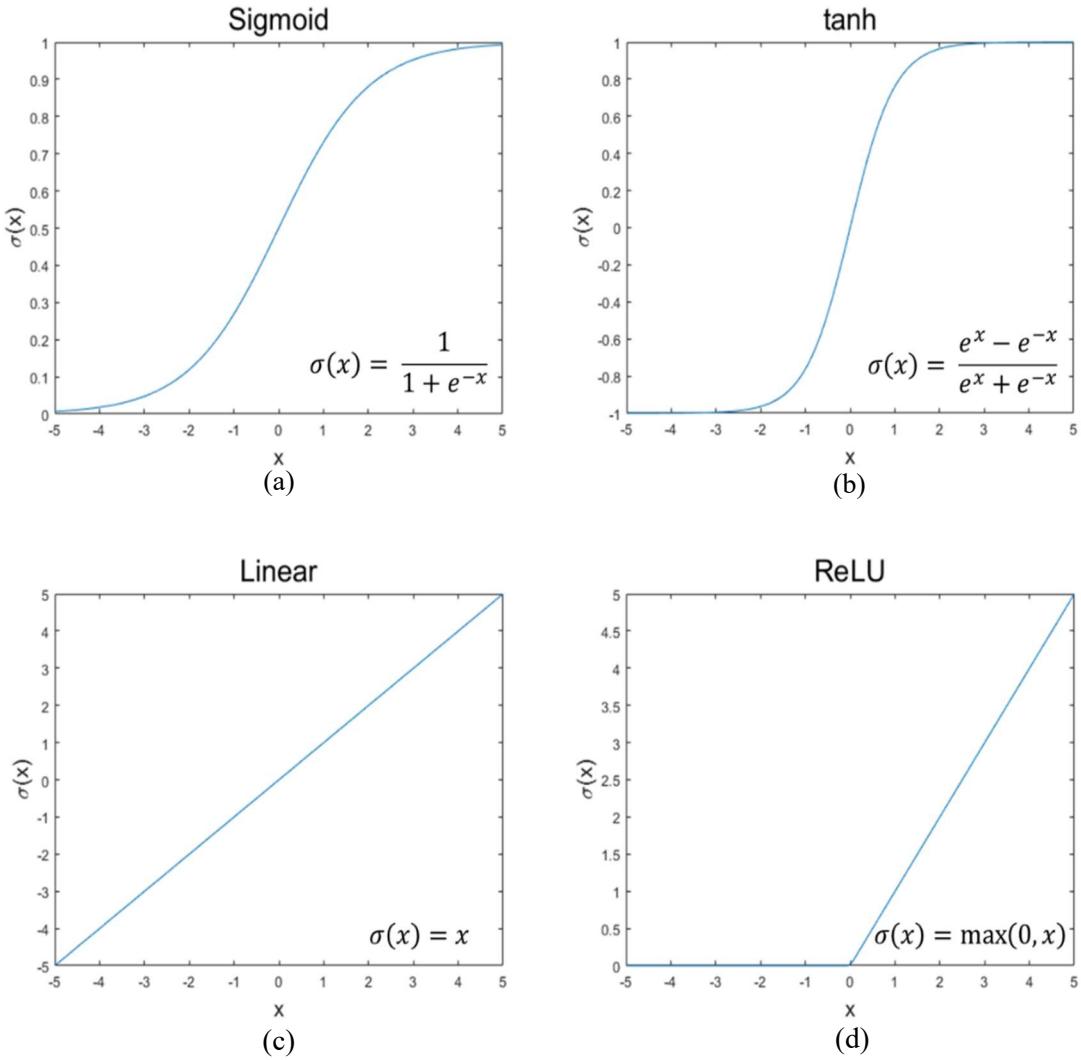


Figure 2.5 Typical activation functions used for neural networks:
(a) sigmoid, (b) hyperbolic tangent, (c) linear, and (d) ReLU.

인공신경망 각 층의 개별 노드는 인접 층의 모든 노드와 가중치로 연결되기 때문에 입력층과 은닉층, 출력층을 모두 아울러서 완전연결층(fully connected layer)이라고 부른다(O`Shea and Nash, 2015). N^{fc} 개의 완전연결층을 가진 인공신경망의 전체 연산 과정을 g 라고 한다면, 인공신경망의 최종 출력값 $\mathrm{Y}_{N^{fc}}$는 식 (1)을 변형하여 다음과 같이 나타낼 수 있다.

$$\mathrm{Y}_{N^{fc}} = g\left(\mathrm{X};\ \theta^{fc}\right) = \sigma(\mathrm{W}^{fc}_{N^{fc}-1}\sigma\left(\cdots\left(\mathrm{W}^{fc}_{2}\sigma\left(\mathrm{W}^{fc}_{1}\mathrm{X}+\mathrm{B}^{fc}_{1}\right)+\mathrm{B}^{fc}_{2}\right)\cdots\right)+\mathrm{B}^{fc}_{N^{fc}-1}\right), \quad (2)$$

이 때 $\theta^{fc} \coloneqq \left(\mathrm{W}^{fc}_{1},\cdots,\mathrm{W}^{fc}_{N^{fc}-1},\mathrm{B}^{fc}_{1},\cdots,\mathrm{B}^{fc}_{N^{fc}-1}\right)$ 는 학습가능 매개변수(trainable parameter)인 가중치 W 와 편향 B 의 집합이며, 윗첨자 fc 는 완전연결(full connection)을 의미한다. 결론적으로 인공신경망의 학습은 최적의 θ^{fc} 를 결정하는 과정이 된다.

최적의 W 와 B 는 앞서 설명한 역전파 기법에 의해 손실 함수 J 를 최소화함으로써 결정된다. L2-norm으로 나타낸 J는 식 (3)과 같다.

$$J = \left\|\mathrm{Y}_{N^{fc}} - \hat{\mathrm{Y}}\right\|_2, \quad (3)$$

이 때 $\hat{\mathrm{Y}}$은 학습 자료 중 출력값(참값)을 의미한다.

2.3. 합성곱 신경망

합성곱 신경망은 대표적인 지도 학습 방식의 딥러닝 기법으로 사물 인식과 음성 인식 등에서 탁월한 성능을 보이고 있다. 합성곱 신경망은 입력값의 특징(feature)을 추출하는 전반부와 추출된 특징을 학습하는 후반부로 구성된다(O`Shea and Nash, 2015). 합성곱 신경망의 전반부에서는 입력값으로 다차원 배열을 통째로 받아들인 후, 컨볼루션(convolution)과 풀링(pooling) 연산을 통해 입력값의 특징을 추출한다. 추출된 다차원 배열의 특징은 준평(flattening) 단계를 거쳐 1차원의 배열로 전환된다. 합성곱 신경망의 후반부는 전환된 1차원 배열을 입력받아 완전연결층에서 학습하는 단계이다. 후반부의 구조와 학습 방식은 기존의 인공신경망과 거의 동일하므로 2.2절의 설명으로 갈음한다. 합성곱 신경망의 전반부 연산 과정은 식 (4)와 같이 나타낼 수 있다.

$$\mathrm{Y} = f(\mathrm{X};\ \theta^{cv}), \tag{4}$$

이 때 $\theta^{cv} := \left(\mathrm{W}_1^{cv},\cdots,\mathrm{W}_{N^{cv}}^{cv},\mathrm{B}_1^{cv},\cdots,\mathrm{B}_{N^{cv}}^{cv}\right)$는 컨볼루션 과정의 학습가능 매개변수인 가중치 W와 편향 B의 집합이다. N^{cv}는 컨볼루션층의 수이며, 윗첨자 cv는 컨볼루션(convolution)을 의미한다. 완전연결층의 수보다 1만큼 적게 연산이 수행되던 완전연결층과는 달리, 컨볼루션층에서는 컨볼루션층의 수만큼 연산이 수행된다.

Figure 2.6은 회귀 모형 합성곱 신경망의 구조에 대한 예시이다. 해당 신경망은 입력값으로 2차원 배열로 저장된 채널 저류층의 유체투과도 분포를 사용하며, 컨볼루션층(convolutional layer, 2.3.1절)과 풀링층(pooling layer, 2.3.2절)이 2번씩 반복되는 전반부와 4개의 완전연결층을 가진 후반부로 구성되었다. 그림에 표현되지는 않았지만 완전연결층에서는 드롭아웃(dropout, 2.3.3절) 기법이 적용되어 학습 성능은 뛰어나지만 예측 성능은 저조한 과적합(overfitting)을 억제한다. 활성화 함수를 사용하지 않는 풀링층을 제외한 각 층의 출력값은 활성화 함수인 ReLU함수를 거쳐서 결정되며, 예외적으로 마지막 완전연결층은 선형(linear) 함수를 활성화 함수로 사용한다. 이것은 연속적인 값을 출력값으로 도출하는 회귀 신경망 모형의 특징이다.

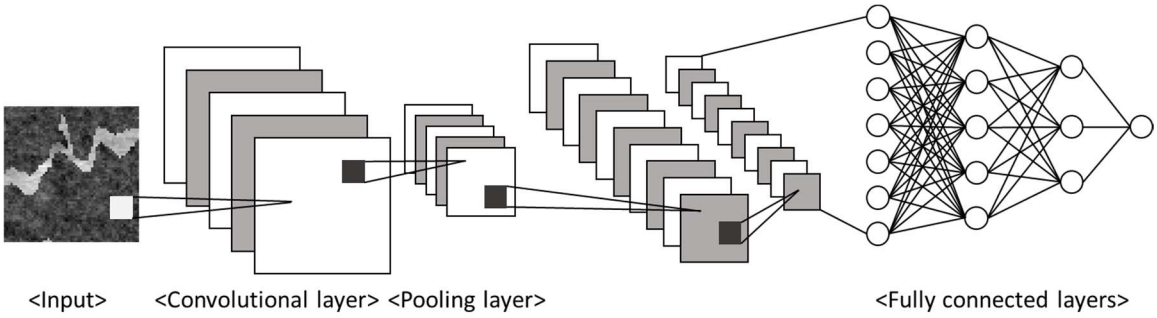


Figure 2.6 Structure of CNN.

2.3.1. 컨볼루션층

컨볼루션층은 커널(kernel)이라고도 불리는 컨볼루션 필터(filter)를 사용하여 입력 자료에서 특징을 추출하여 특징 지도(feature map)를 도출하는 역할을 한다. 컨볼루션 필터는 컨볼루션 연산을 통해 특정한 특징이나 패턴이 입력 자료에 존재하는지 여부를 검출하는 다차원 배열이다. 일반적으로 5×5×5 또는 3×3과 같은 정방 형태의 필터가 많이 사용되지만, 필터의 형태와 개수는 보통 사용자의 경험에 근거한 주관적 판단에 의해 결정된다. 컨볼루션 필터는 필터와 동일한 크기의 영역을 입력 자료에서 선택한 후 요소별 곱셈(element-wise multiplication)을 수행한 후 모두 더하여 하나의 값을 도출한다. 이렇게 도출된 값의 크기가 해당 영역의 특정 필터에 대한 세기이며, 필터는 입력 자료의 전체 영역에 대해서 마치 창문이 움직이듯 동일한 작업을 반복적으로 수행한다. 보통 좌측 상단에서 우측 하단까지 순차적으로 동일 과정을 반복한다. 필터 배열 내부의 각각의 가중치 값은 완전연결층의 가중치 값과 마찬가지로 역전파 기법을 통해 손실 함수를 줄이는 방향으로 갱신된다.

Figure 2.7은 6×5 크기의 입력값 X 에 3×2 크기의 컨볼루션 필터 W 를 적용하여 4×4 크기의 특징 지도 Y를 출력하는 컨볼루션 연산 과정을 나타낸다. 특징 지도의 음수값은 ReLU 함수(Figure 2.5(d))를 거치면서 모두 0으로 바뀌게 된다. 특징 지도의 크기는 입력값과 필터의 크기 외에도 스트라이드(stride) 값에 영향을 받는다. 스트라이드는 필터를 적용하는 간격을 의미하며, 일반적으로 1의 값을 사용한다. Figure 2.7의 연산 과정에서도 1의 스트라이드가 사용되었다.

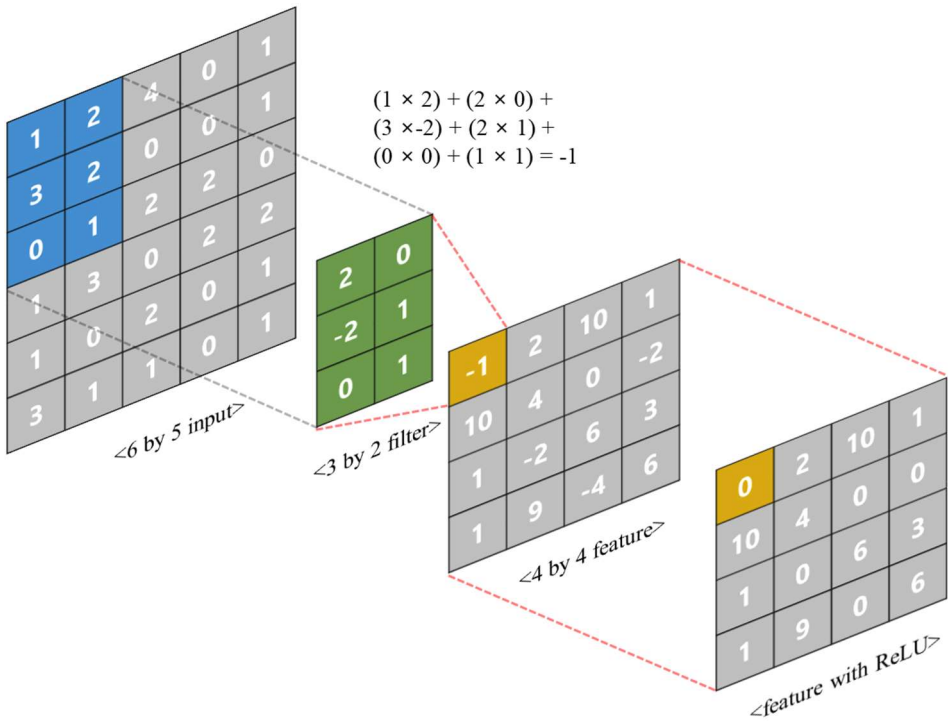


Figure 2.7 Schematic of convolution for a 2D data array.

제로패딩(zero-padding)은 필터를 적용하기 전에 입력값 주위에 0의 값을 둘러서 입력 배열의 크기를 키우는 작업을 뜻한다. 적절한 크기의 패딩을 적용하면 필터를 적용한 후에도 자료의 크기가 유지된다. 여러 단계에 걸쳐 컨볼루션 필터를 적용하면, 출력값의 크기가 너무 작아져 분석이 어렵거나 특징이 유실될 수 있기 때문에 이러한 상황을 방지하기 위해 제로패딩을 적용한다.

Figure 2.8은 5×5 크기의 입력값에 폭이 1인 제로패딩을 적용한 경우와 적용하지 않은 경우의 출력되는 특징 지도의 크기 차이를 나타낸다. 패딩을 적용하지 않으면 출력값의 크기가 3×3으로 줄어들지만, 제로패딩을 적용하면 5×5로 유지된다.

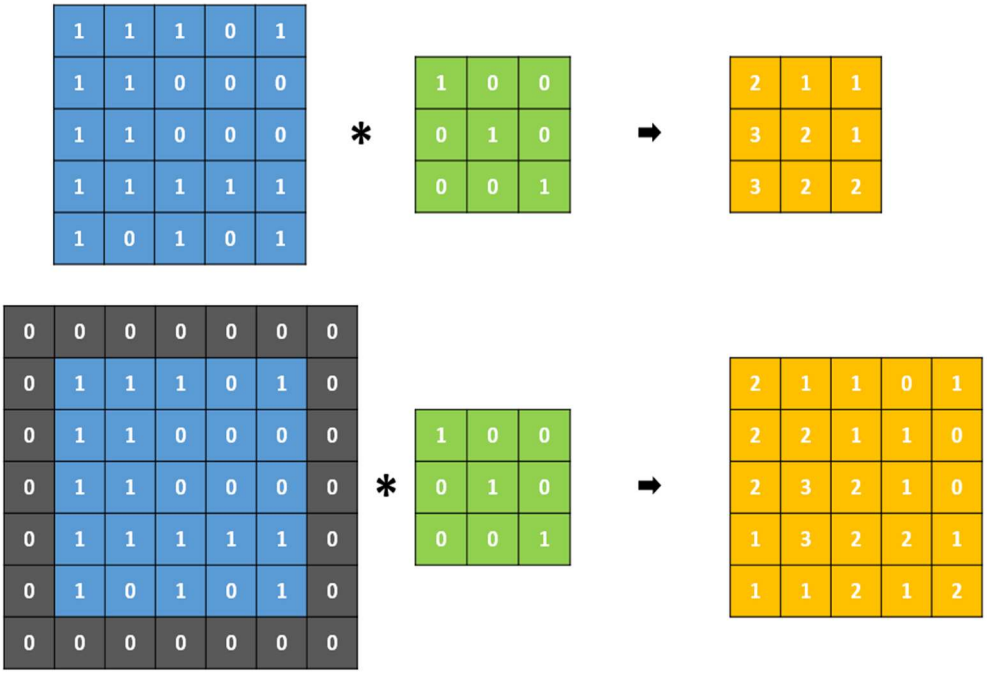


Figure 2.8 Schematic of zero-padding.

2.3.2. 풀링층

풀링층은 입력값을 여러 하위 구역(sub-region)으로 나누고 각각의 하위 구역을 하나의 대푯값으로 축소하여 자료의 크기를 줄이고 특징을 강조하는 역할을 한다. 풀링의 적용 영역은 컨볼루션과는 달리 서로 겹치지 않도록 조정하기 때문에 출력값의 크기가 대폭적으로 줄어들게 된다. 해당 과정은 배열 내 요소의 수를 줄인다는 점에서 석유공학에서의 업스케일링(upscaling) 과정과 상당히 유사하다. 주로 각 구역에서 최댓값을 대푯값으로 도출하는 맥스풀링(max pooling) 기법이 사용된다. Figure 2.9는 Figure 2.7의 컨볼루션 과정에서 도출된 특징 지도에 2×2 크기의 맥스풀링을 적용하여 2×2 크기의 출력값을 도출하는 과정을 나타낸다.

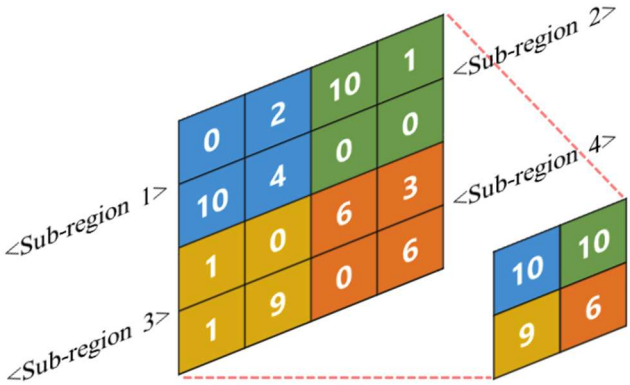


Figure 2.9 Schematic of max pooling for a 2D data array.

2.3.3. 드롭아웃

드롭아웃(dropout) 기법은 학습 자료의 특성을 과도하게 반영함으로써 모델이 지나치게 복잡해지는 과적합 문제를 해결하기 위해 제안되었다(Hinton et al., 2012). 과적합이 발생하면 학습되지 않은 자료에 대한 모델의 예측 성능이 감소하여 범용성이 떨어지게 된다. 이러한 현상을 방지하기 위해 여러 딥러닝 기법에서는 학습 과정에서 완전연결층의 노드 전체를 사용하지 않고, 일부 노드를 비활성화 하여 해당 노드에 연결된 가중치를 학습에서 배제하는 드롭아웃을 적용한다.

Figure 2.10은 Figure 2.4의 인공신경망의 은닉층에 50%의 비율로 드롭아웃을 적용한 경우를 나타낸다. 드롭아웃이 적용되면 학습에 사용되는 가중치의 수가 대폭적으로 감소한다. 또한, 학습이 반복될 때마다 비활성화 되는 노드를 랜덤하게 바꾸게 되므로 매번 서로 다른 신경망을 사용하는 효과를 가지게 되고, 결과적으로 과적합을 방지한다.

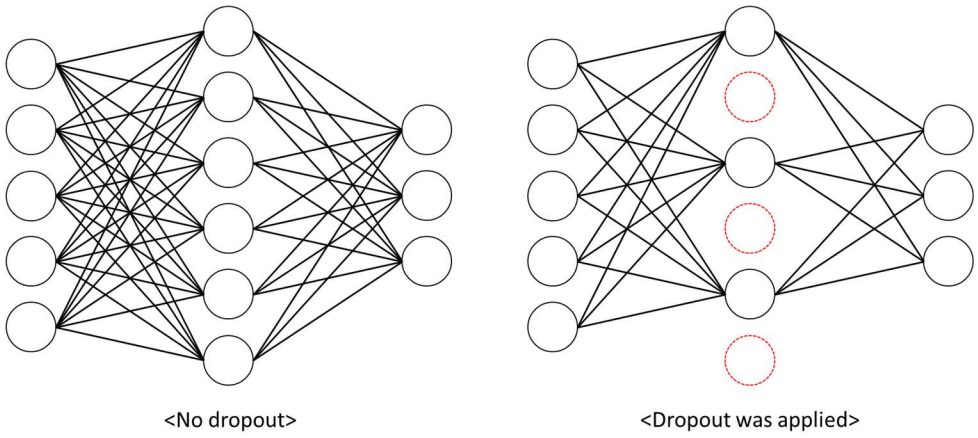


Figure 2.10 Schematic of dropout.

3. 딥러닝 프록시를 사용한 추가정의 최적 위치 선정

3장에서는 이 연구에서 제안한 딥러닝 프록시 기법인 다중모달 학습이 적용된 합성곱 신경망 기법으로 추가정의 최적 위치 선정에 대한 과정 및 개발 환경을 설명한다. 제안한 기법은 유정 인근의 정적(유체투과도, 공극률) 및 동적(압력, 오일 포화도) 물성과 기타 정보(유정의 위치에 기반한 거리 정보)를 바탕으로 유정의 생산성을 평가한다. 제안한 기법으로 선별한 생산성이 높은 지점에 대해서 저류층 시뮬레이션을 수행하여 검증한 후 최적의 시추 위치를 선정하게 된다. 제안한 기법은 파이썬(Python) 기반의 딥러닝 라이브러리 케라스(Keras)와 구글(Google)의 텐서플로(TensorFlow)를 사용하여 구현하였다.

3.1. 다중모달 합성곱 신경망

사물인식 등에 사용되어 온 전형적인 합성곱 신경망은 다차원 배열의 특징을 추출하고 학습할 수 있지만, 한 종류의 입력 자료만을 사용하는 한계가 있다. 유정의 생산성에 영향을 미치는 저류층 인자의 수는 매우 많으며, 그 종류나 양식 또한 다양하다. 따라서 다양한 입력 자료를 동시에 고려하여 유정의 생산성에 대한 예측 성능을 높이기 위하여 다중모달 학습을 합성곱 신경망에 적용하였다. 다중모달 학습이란 1장에서 설명했듯이 다양한 양식의 자료를 통합하여 모델링하는 것을 의미한다. 다중모달 합성곱 신경망은 입력 자료의 특징을 추출하는 합성곱 신경망의 전반부(컨볼루션층과 풀링층)를 학습

자료의 종류만큼 병렬적으로 연결한 거대한 규모의 신경망이다(Eitel et al., 2015). 이 논문에서는 병렬적으로 연결된 각각의 합성곱 신경망의 전반부를 컨볼루션 분기(convolution branch)로 명명한다. 개별 컨볼루션 분기에서 추출된 특징은 각각 준평 단계를 거쳐 1차원 배열로 바뀐 후, 연결(concatenating) 단계에서 하나의 1차원 배열로 합쳐진다. 유정과 저류층 경계까지의 거리와 같은 다차원 배열로 나타낼 수 없는 1차원 배열의 정보는 연결 단계에서 추가되어 완전연결층으로 입력한다.

i 번째 컨볼루션 분기를 $f_i(\mathrm{X}_i;\ \theta_i^{cv})$ 라고 할 때, 각각의 컨볼루션 분기의 출력값은 준평 단계를 거쳐 1차원 배열로 변환되고 연결 함수(concatenate function) h에 의해 하나의 1차원 배열로 합쳐진다. 이 때 함수 h에는 거리 정보 D가 추가적으로 입력된다. 해당 과정을 수식으로 나타내면 다음과 같다.

$$\mathrm{Y} = h([f_1(\mathrm{X}_1;\ \theta_1^{cv}),\cdots,f_I(\mathrm{X}_I;\ \theta_I^{cv}),\mathrm{D}]), \tag{5}$$

이 때 I는 컨볼루션 분기의 수를 의미한다. 이 논문에서는 정적 물성 두 종류와 동적 물성 두 종류, 최대 네 종류(유체투과도, 공극률, 압력, 오일 포화도)의 입력값을 사용한다. 유체투과도와 공극률은 각각 암석에서의 유체의 유동 능력과 유체의 저장 능력을 의미한다. 압력과 오일 포화도 역시 유사한 의미를 가진다. 동적 물성은 시간에 따라 변화하기 때문에 추가정이 추가되는 시점을 기준으로 저류층 시뮬레이션에 기반하여 획득하였다. 정적 물성과 동적 물성은 추가정을 중심으로 유정 주위의 영역에서 획득하며, 해당 영역은 유정의 배유

반경을 고려하여 결정한다. 해당 영역 내부의 격자에서 추출한 물성값을 각각 3차원 배열로 저장한다. 거리 정보 D는 기존 생산정과 추가정 사이의 거리 및 추가정과 저류층 사방 경계까지의 거리로 구성되며, 1차원 배열로 저장한다.

다중모달 합성곱 신경망의 전체 연산 과정은 식 (6)과 같이 나타낼 수 있다.

$$\mathrm{Y}_{N^{fc}} = g(h([f_1(\mathrm{X}_1;\ \theta_1^{cv}),\cdots,f_I(\mathrm{X}_I;\ \theta_I^{cv}),\mathrm{D}]);\ \theta^{fc}). \tag{6}$$

해당 신경망은 손실 함수(식 (3))를 최소화하기 위하여 θ^{cv}와 θ^{fc}를 갱신하는 과정을 통해서 학습된다. 학습된 신경망 모델은 모든 후보 지점에 대한 추가정의 생산성을 신속히 예측하는데 사용한다.

Figure 3.1은 두 개의 컨볼루션 분기를 가지는 다중모달 합성곱 신경망의 구조를 나타낸다. 각각의 컨볼루션 분기에는 3차원 배열로 저장된 입력값이 한 종류씩 입력된다. 1차원 배열로 저장된 거리 정보는 연결 단계에서 추가되어 완전연결층에 직접 입력된다. 개별 층의 활성화 함수로 ReLU 함수를 사용하며, 마지막 완전연결층에서만 선형(linear) 활성화 함수를 사용한다. 그림에는 표현되지 않았지만 완전연결층에서는 드롭아웃이 적용되어 과적합을 방지한다.

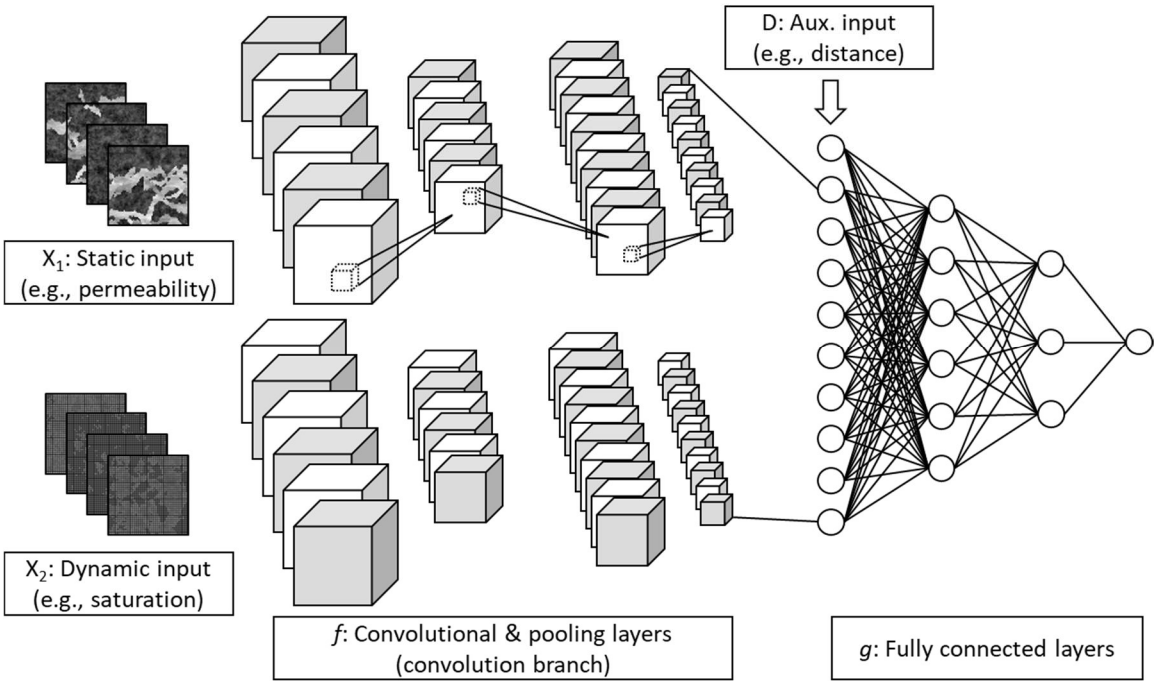


Figure 3.1 Structure of the proposed multi-modal CNN.

3.2. 다중모달 합성곱 신경망을 사용한 연구의 과정

Figure 3.2는 다중모달 합성곱 신경망을 사용한 추가정의 최적 위치 선정 과정을 나타낸 순서도이다. 이 연구에서는 유정 인근의 격자별 물성 분포와 유정의 생산성 사이의 관계를 합성곱 신경망으로 학습한 후, 이를 바탕으로 추가정의 최적 위치를 선정하고자 하였다. 먼저 연구자가 선택한 위치나 임의의 위치에 대해서 저류층 시뮬레이션을 수행하여 학습 자료를 획득한다. 학습자료 획득의 사례는 4.1절과 4.2절에 기술하였다. 다음으로 합성곱 신경망의 구조를 설계한다. 신경망의 구조에 영향을 미치는 요소를 초매개변수(hyper-parameter)라고 하며, 이에 대해서는 3.3절에서 자세히 설명한다. 신경망 구조 설계의 사례는 4.3절에 기술되어 있다. 학습 자료를 학습 세트(training set), 검증 세트(validation set), 시험 세트(test set)로 나눈 후 신경망의 학습을 진행한다. 실제 학습은 학습 세트만을 사용해서 이루어지며, 검증 세트는 학습 도중에 모델의 성능을 확인하기 위해 사용한다. 학습이 종료되면 시험 세트를 사용해 최종적으로 모델의 성능을 검증한다. 모델의 성능이 기준을 만족하는 경우, 학습된 합성곱 신경망을 사용해 모든 후보 지점에 대해 생산성을 예측한다. 이 연구에서 사용하는 기준은 손실 함수 J와 결정계수(coefficient of determination) R^2이다. 합성곱 신경망의 예측 결과를 내림차순으로 정리한 후, 생산성이 높은 지점과 중간 지점, 낮은 지점에 대해서 저류층 시뮬레이션을 수행해 생산성을 비교함으로써 두 번째 검증을 실시한다. 두 번째 검증의 기준은 신경망의

출력값과 시뮬레이션의 결과값 사이의 오차 ε이다. 두 번째 검증의 결과가 좋은 경우, 생산성이 가장 높게 예측된 지점들에 대해서 저류층 시뮬레이션을 수행한 후 최종적으로 저류층 시뮬레이션의 결과에 기반해 최적의 위치를 선정한다. 모델의 성능이 기준을 만족하지 못하는 경우에는 초매개변수를 조절하여 합성곱 신경망의 구조를 변경한 후 학습을 다시 진행한다. 학습된 신경망을 사용한 최적 시추위치 선정 사례는 4.4절과 4.5절에 기술하였다.

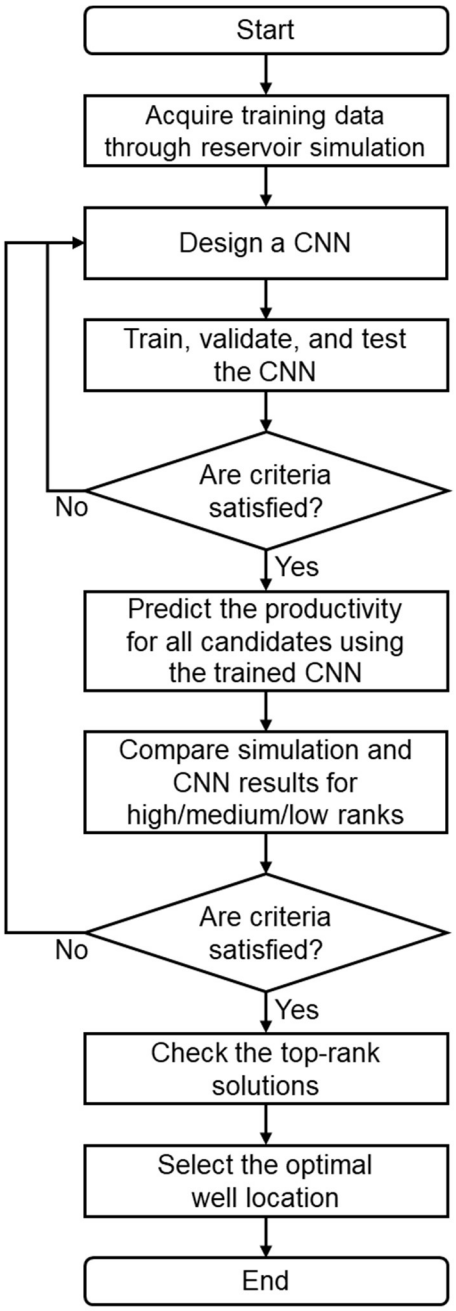


Figure 3.2 Flowchart to show how to select the optimal infill well placement using the multi-modal CNN.

3.3. 합성곱 신경망의 초매개변수

신경망 모델을 구성하는 요소는 크게 학습 과정에서 값이 결정되는 학습가능 매개변수와 연구자가 직접 결정하는 초매개변수로 나뉜다. 가중치와 편향은 대표적인 학습가능 매개변수로 이들은 학습 과정에서 조금씩 값이 변화하며 최적의 값으로 결정된다. 반면, 초매개변수는 신경망의 구조를 결정하는 요소로 연구자가 수정하기 전까지는 그 값이 변하지 않는다. 이러한 초매개변수의 종류는 매우 다양하며, 신경망의 성능에 미치는 영향이 각기 다르기 때문에 신경망을 사용하는 연구에 있어서 최적의 초매개변수 디자인은 필수적이다. 최근에는 기계가 개별 초매개변수의 영향력을 분석하고 최적의 디자인까지 도출하는 자동 기계 학습(automated machine learning) 분야의 연구도 활발하다(Feurer et al., 2015).

Table 3.1은 합성곱 신경망의 초매개변수의 종류를 나타낸다. 초매개변수는 학습 모델, 계층(layer), 학습 자료와 관련된 부류로 나눌 수 있다. 초매개변수의 종류는 매우 다양하며 딥러닝 기법을 적용하는 문제에 따라 각각의 초매개변수의 영향력이 달라지기 때문에 모든 초매개변수를 다루는 것은 매우 어렵다. 이 논문에서는 다양한 종류의 다차원 배열의 입력값을 통합하는 효과에 집중하기 위하여 컨볼루션 필터의 수와 크기, 컨볼루션층의 수를 주로 수정하며 최적의 신경망 구조를 결정하였다.

Table 3.1 List of hyper-parameters

Neural network model	Layer	Training data
Types of model Epoch Learning rate Loss functions Optimizing methods Size of batch etc.	Number of hidden layers Number of nodes Number of convolutional layers Number of convolutional filters Size of convolutional filter Number of pooling layers Size of pooling Types of pooling Zero-padding Activation functions Initializing methods Dropout rate etc.	Types of input data Size of input data Normalizing methods etc.

3.4. 딥러닝 프록시 모델의 개발 환경

이 절에서는 제안한 다중모달 합성곱 신경망의 개발 환경을 기술한다. 이 연구는 케라스(Keras)와 구글 텐서플로(TensorFlow)를 사용하여 딥러닝 모델을 구현하였다. 구체적으로 프로그램 코딩에는 케라스를 사용하였으며, 실제 연산은 텐서플로로 수행하였다. 이는 실제 연산을 백엔드(back-end) 엔진에서 수행하도록 하는 케라스의 특징에서 비롯된다. 백엔드란 컴퓨터 시스템의 데이터 처리 부분을 의미한다.

Table 3.2는 2017년 11월을 기준으로 온라인 오픈소스 저장소 GitHub, 개발자 커뮤니티 stack overflow, 구글 검색 결과를 통합하여 나타낸 딥러닝 라이브러리 리스트이다. 구글 텐서플로가 가장 널리 사용되고 있는 가운데 케라스와 카페(Caffe) 등의 딥러닝 라이브러리가 사용되고 있다. 이러한 딥러닝 라이브러리는 대부분 프로그래밍 언어 파이썬(Python)을 기반으로 개발되었으며, GPU(graphic processing unit) 연산 제어와 같은 기능을 제공함으로써 빅데이터를 주로 다루는 딥러닝 연구에 특화되어 있다.

Table 3.2 List of deep learning libraries (https://www.techleer.com)

Rank	Deep learning library	Overall
1	TensorFlow	10.87
2	Keras	1.93
3	Caffe	1.86
4	Theano	0.76
5	Pytorch	0.48

케라스는 파이썬으로 구현된 상위 레벨의 오픈소스 딥러닝 라이브러리이다. Theano, 구글 텐서플로, 마이크로소프트 CNTK등의 다른 딥러닝 라이브러리를 백엔드로 연동하여 연산 엔진으로 사용한다. 케라스는 하위 레벨의 계산을 직접 수행하지 않는 대신 상위 레벨의 사용자 친화적이고 직관적인 인터페이스를 제공한다. 따라서 동일한 내용의 코드 작성 시 텐서플로와 같은 다른 딥러닝 라이브러리에 비해 코드가 훨씬 짧고 직관적인 특징이 있다(Chollet, 2017). 케라스에서 제공하는 기능은 대부분 독립적인 모듈로 구성되어 확장성이 우수하며 간결하다. 신경망 계층, 손실 함수, 활성화 함수, 초기화 기법, 최적화 기법 등이 모두 독립적인 모듈로 구현되어 있으므로 사용자는 필요한 모듈을 조합하여 원하는 딥러닝 모델을 구축할 수 있다.

Table 3.3은 텐서플로와 케라스의 장점과 단점을 비교한 표이다. 전반적으로 텐서플로는 정교한 모델을 구현할 수 있으나 사용하기가 어렵고, 케라스는 사용하기가 쉬우나 매우 정교한 모델의 구현이 다소 어렵다는 특징을 가진다.

Table 3.3 Comparison between TensorFlow and Keras

	TensorFlow	Keras
Advantage	- Fast processing speed - More relevant libraries and functionalities	- Easy to code - Various back-end engines
Disadvantage	- Too complicated to make source codes	- Relatively slow processing speed - Dependent on back-end engine - Less relevant libraries and functionalities

4. 연구 결과

4장은 다중모달 합성곱 신경망 모델을 이용한 가상 저류층에서의 추가정의 최적 위치 선정을 다룬다. 다중모달 합성곱 신경망 모델의 성능은 기존의 인공신경망의 성능과 비교 및 분석하였다. SPE10 저류층 모델을 합성곱 신경망의 학습, 검증, 시험 자료를 획득하기 위한 저류층 시뮬레이션의 대상으로 사용하였다.

4.1. 저류층 모델

SPE10 저류층 모델은 영국과 노르웨이 사이에 위치한 북해(North Sea)의 지질정보에 기반하여 제작된 가상의 저류층 모델이다. 북해의 브렌트 층군(Brent group)은 노르웨이 앞바다의 East Shetland 분지에 위치하며 쥐라기 중기에 형성된 다섯 개의 지층으로 구성된다. SPE10 저류층 모델은 다섯 개의 지층 중 Tarbert 층과 Ness 층의 실제 지질 정보에 기초하여 10th SPE Comparative Solution Project에서 개발되었으며, 수공법(waterflooding) 과정의 업스케일링 연구의 벤치마크 모델로 주로 사용되었다(Christie and Blunt, 2001).

SPE10 모델은 총 85개 층으로 구성된다. 하부 50개 층은 삼각주(delta)에서 퇴적된 Ness 층의 채널 환경을 모사하고, 상부 35개 층은 삼각주보다 다소 깊은 근해(shore) 환경에서 퇴적된 Tarbert 층을 모사한다. 일반적으로 삼각주나 곡류(meander)와 같은 하천(fluvial) 환경에서 퇴적된 지층은 하천의 형태가

투사된 뚜렷한 채널 패턴을 보인다. 이러한 채널 내부의 암석은 비교적 큰 입자로 구성되며, 채널 외부의 암석은 입자의 크기가 작다. 따라서 채널 내부에 유체투과도가 높은 우수한 저류층이 형성되는 경우가 많다. 반면, 비교적 안정적인 근해 환경에서 형성된 지층은 일반적으로 해안에 가까울수록 이동거리가 짧은 큰 입자가 분포하고 해안에서 멀어질수록 이동거리가 긴 작은 입자가 분포하는 경향을 보인다. 그러나 입자 크기의 분포가 채널 저류층에 비해서 훨씬 연속적이므로 불연속적인 경계면은 잘 나타나지 않는다(Hyne, 2001). Ness 층을 모사한 SPE10 모델의 하부는 매우 뚜렷한 채널 패턴이 나타난다. 채널 내부는 유체투과도와 공극률이 우수한 사암으로 이루어지고, 채널 외부는 머드가 퇴적된 셰일로 구성되어 유체투과도와 공극률이 매우 낮은 특징을 가진다. 근해 환경에서 퇴적된 Tarbert 층을 모사한 SPE10 모델의 상부는 하부와 달리 불연속적인 경계면이 나타나지 않으며 물성의 분포가 매우 부드럽고 연속적인 특징을 보인다. 이 논문에서는 다수의 학습 자료를 빠르게 획득하고자 상부와 하부의 상위 5개 층씩을 개별적인 부분 저류층으로 구성하여 두 가지의 사례 연구를 진행하였다. 앞으로 이러한 부분 저류층 모델을 각각 근해 저류층과 채널 저류층으로 명명한다.

Figure 4.1은 각각의 저류층 모델의 두 번째 층의 유체투과도 분포를 나타낸다. 각 저류층 모델은 66,000(=220×60×5)개의 격자로 구성되며, 개별 격자의 크기는 50×50×10 ft^3이다. 최상부층의 심도는 12,000 ft이며, 저류층의 초기 압력은 6,000 psi, 초기 오일 포화도는 약 0.80으로 설정하였다. 또한,

최하부층의 하부에 대수층을 연결하여 저류층의 하부에서부터 물이 유입되도록 설정하였다.

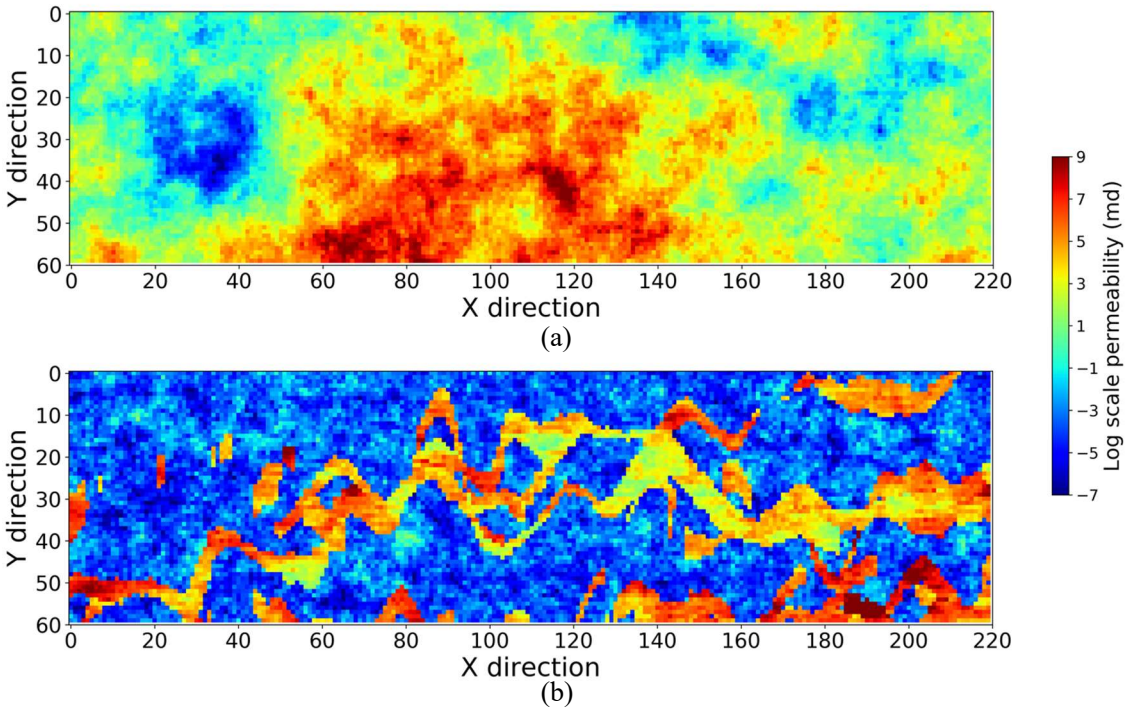


Figure 4.1 Permeability distribution of the SPE10 model: (a) the 2nd layer of the shoreface reservoir and (b) the 2nd layer of the channelized reservoir.

Table 4.1은 연구에서 사용한 SPE10 부분 모델의 유체투과도와 공극률 범위를 나타낸다. 원본 자료에는 공극률이 0인 격자가 다수 존재하였으나, 0.01 미만인 격자의 공극률을 모두 0.01로 수정하여 사용하였다.

Table 4.1 Distribution of permeability and porosity for the SPE10 reservoir model

Reservoir properties	max	min	mean	median	std.
Permeability in the shoreface reservoir (md)	20,000	7.13E-04	403.217	19.9674	1,822
Permeability in the channelized reservoir (md)	20,000	9.82E-04	230.772	0.2569	1,134
Porosity in the shoreface reservoir (fraction)	0.5	0.01	0.1785	0.1787	0.086
Porosity in the channelized reservoir (fraction)	0.4	0.01	0.1265	0.1019	0.093

Figure 4.2는 연구에서 사용한 SPE10 저류층 모델의 상대 유체투과도 곡선을 나타내며, Table 4.2는 연구에서 사용한 유체의 주요 물성을 나타낸다.

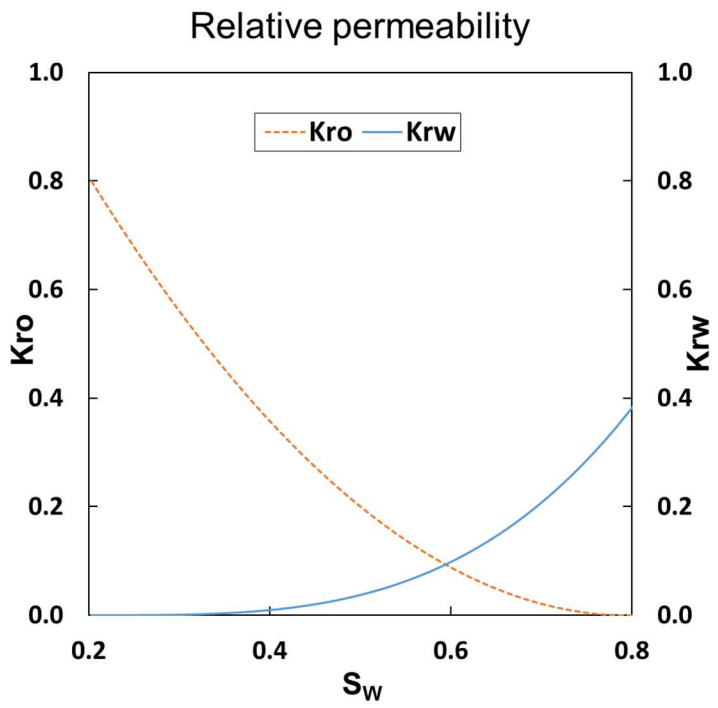
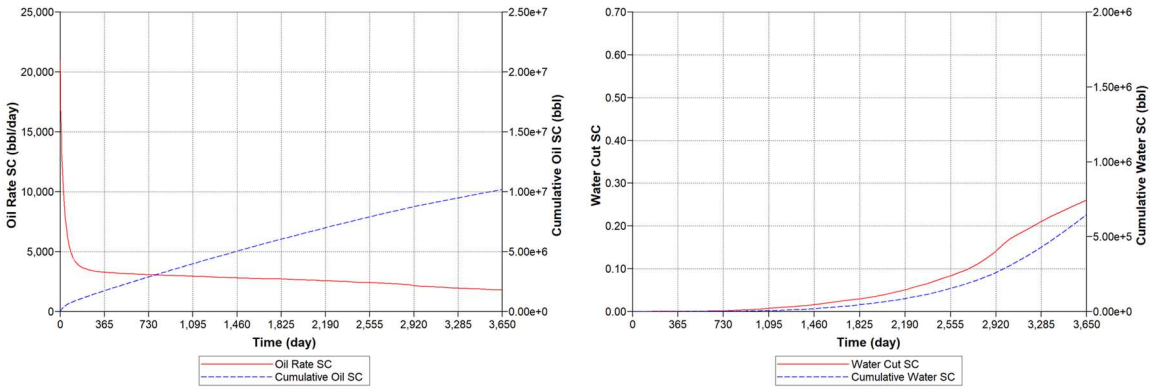


Figure 4.2 Relative permeability curves.

Table 4.2 Properties of fluids

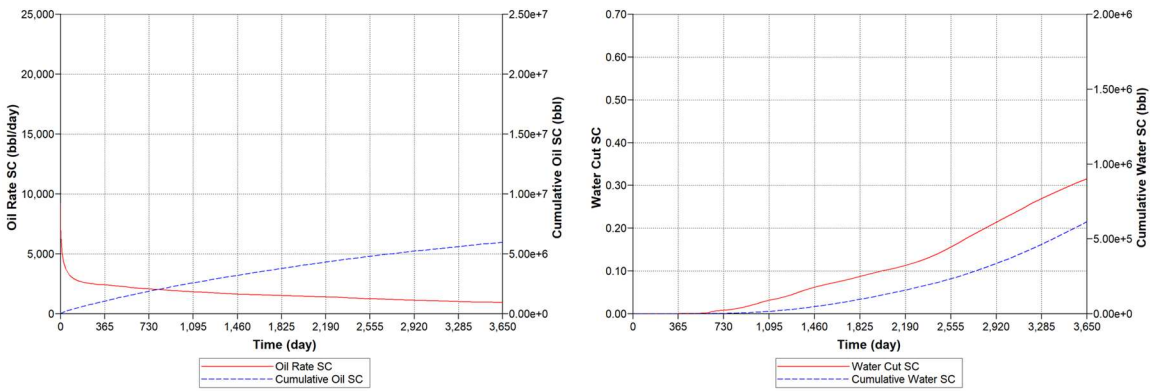
Properties	Value
Oil viscosity @ SC (cp)	1.04
Water viscosity @ SC (cp)	0.31
Oil density @ SC (lb/ft^3)	46.244
Water density @ SC (lb/ft^3)	62.419
Oil compressibility (psi^{-1})	5×10^{-6}
Water compressibility (psi^{-1})	3.15×10^{-5}

SPE10의 근해와 채널 각각의 저류층 모델에는 정중앙인 (110, 30) 지점에 기존 생산정 P1이 위치한다. 기존 생산정에서 10년 동안의 생산이 진행된 후, 임의의 위치에 추가정 P2가 배치되어 추가적으로 10년 동안 생산이 진행된다. Figure 4.3과 Figure 4.4는 근해와 채널 저류층의 기존 생산정의 생산 프로필을 나타낸다. 모든 생산정의 공저압력은 4,000 psi를 유지하도록 하였다. 저류층 모델에 대한 보다 자세한 내용은 CMG-IMEX 예제를 첨부한 부록 A를 참조할 수 있다.



(a) (b)

Figure 4.3 Production profile of the existing well at the SPE10 shoreface reservoir over ten years: (a) oil production rate and cumulative oil production and (b) watercut and cumulative water production.



(a) (b)

Figure 4.4 Production profile of the existing well at the SPE10 channelized reservoir over ten years: (a) oil production rate and cumulative oil production and (b) watercut and cumulative water production.

4.2. 딥러닝 프록시 모델의 학습 자료

지도 학습에서 학습 자료는 입력값과 출력값(참값)의 쌍으로 이루어진다. 참값은 회귀 문제에서는 연속된 값으로, 분류 문제에서는 분절되어 있는 클래스의 형태로 나타낸다. 프록시 모델의 설계가 적절하다면, 학습이 진행될수록 참값과 실제 모델의 출력값 사이의 오차가 줄어들게 된다. 이 연구는 추가 생산정 인근의 물성 분포로부터 해당 시나리오의 생산성을 예측하여 최적의 추가정의 위치를 도출하는 것을 목표로 하고 있기 때문에 회귀 모델을 사용하여 구체적인 생산량을 구하였다.

추가정을 중심으로 유체투과도와 같은 정적 물성값은 배유 반경을 고려하여 21×21×5와 같은 3차원 배열로 저장하고, 추가정과 기존 생산정 사이의 거리 및 추가정에서 저류층 경계까지의 거리를 5×1의 1차원 배열로 저장하여 입력 자료로 사용하였다. 오일 포화도, 압력과 같은 동적 물성 배열은 초기 10년의 생산이 끝난 시점을 기준으로 획득하였으며, 정적 물성 배열과 동일한 크기를 사용하였다. 모든 입력값은 표준화(standardization)를 거쳐 평균이 0이고 표준편차가 1인 동일한 스케일로 변형하여 사용하였다(식 (7)). 출력값으로는 추가정의 10년 동안의 누적 오일 생산량을 사용하였다. 학습 자료의 출력값(참값)의 범위는 근해 저류층은 [42975, 4893080]이며, 채널 저류층은 [42027, 1999180]이다.

$$X' = (X - \mu)/\sigma_{std}, \tag{7}$$

이 때 X는 원본 입력값이며, X'은 표준화를 통해 변형된 입력값이다. μ와 σ_{std}는 각각 평균과 표준편차를 의미한다.

학습에 사용한 모든 시뮬레이션 자료는 CMG-IMEX를 실행하여 획득하였다. 총 획득 자료는 근해 저류층은 1,000개, 채널 저류층은 504개이다. 근해 저류층의 초기 132개의 자료는 일정한 간격으로 획득하였으며, 나머지 868개의 자료는 임의의 위치에서 획득하였다. 총 획득가능한 학습 자료의 수가 13,200(=220×60)개 이므로, 학습 자료의 획득 비율은 약 7.6%이다. 300개의 자료는 시험 세트로 사용하였고, 나머지 700개 중 20%인 140개는 검증 세트로, 560개는 학습 세트로 사용하였다. 즉, 학습 세트와 검증 세트, 시험 세트의 비율은 560:140:300이다. 채널 저류층의 경우, 현실성을 위하여 추가정이 채널 내부에 존재하는 경우만 선별하여 504개의 학습 자료를 획득하였다. 학습 자료의 획득 비율은 약 3.8%이며, 학습 세트와 검증 세트, 시험 세트의 비율은 320:80:104이다.

4.3. 딥러닝 프록시 모델의 구조

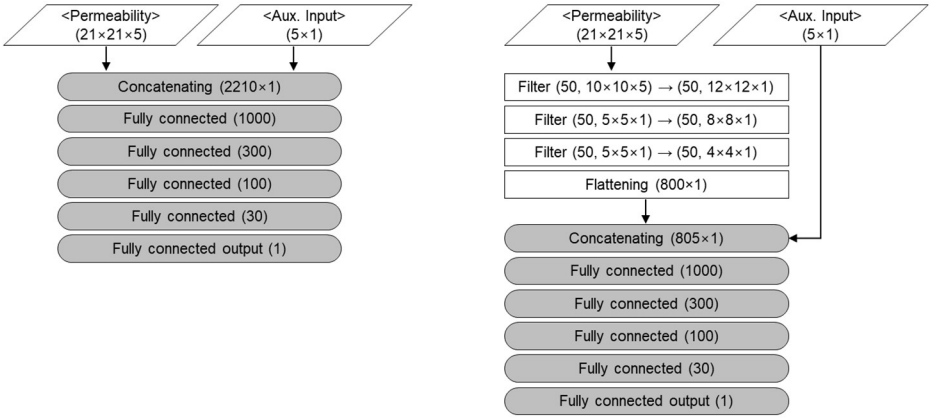
Figure 4.5는 이 연구에서 사용한 근해 저류층에서의 인공신경망과 싱글모달(single-modal) 합성곱 신경망의 구조를 나타낸다. 완전연결층 부분은 회색의 타원으로, 컨볼루션 분기 부분은 흰색의 네모로 나타내었다. 인공신경망에서는 입력값은 곧바로 입력층(input layer)의 역할을 수행하는 연결(concatenating) 단계를 거쳐 완전연결층으로 전달된다. 연결 단계의 노드 수는 2,210개이며, 이것은 유체투과도 입력값인 2,205(=21×21×5)와 거리 정보 입력값인 5가 합쳐진 값이다(Figure 4.5(a)). 반면에 Figure 4.5(b)의 합성곱 신경망에서는 컨볼루션 분기에서 세 개의 컨볼루션층을 거친 후 준평(flattening) 단계를 거쳐 연결 단계에 도달한다. 연결 단계 이후의 완전연결층 단계는 인공신경망과 동일한 구조를 갖도록 설정하였다. 개별 컨볼루션층에서의 연산은 컨볼루션층 내부에 표기하였다. 컨볼루션층 내부의 화살표(→) 왼쪽 항목은 컨볼루션 필터의 정보를 나타내고, 오른쪽 항목은 출력값의 정보를 나타낸다. 즉, 첫 번째 컨볼루션층에서는 21×21×5의 입력값이 입력되고 50개의 10×10×5 크기의 필터가 적용되어 50개의 12×12×1의 출력값이 도출된다. 이 출력값은 다음 층의 입력값으로 사용한다. 컨볼루션층의 입력값과 출력값은 공간적 차원과 스펙트럼 차원을 가진다. 예를 들어 400(=20×20) 픽셀의 RGB 이미지의 경우, 공간적 차원은 이미지의 크기인 20×20이며, Red, Green, Blue 채널이 모여 하나의 이미지를 나타내기 때문에 스펙트럼 차원은 3이 된다. 이 연구에서

입력값으로 사용하는 유체투과도와 같은 물성은 그레이 스케일과 같이 채널이 하나이므로 1의 스펙트럼 차원을 가진다. 출력값의 경우 2.3.1절에서 설명한 과정을 거쳐 공간적 차원이 결정되며, 해당 층의 컨볼루션 필터의 수만큼 스펙트럼 차원을 가지게 된다. 세 번째 컨볼루션층의 출력값은 4×4×1의 공간적 차원과 50의 스펙트럼 차원을 가지므로 준평 단계에서 800×1의 1차원 배열로 변환된다(800=4×4×1×50). 듀얼모달(dual-modal) 합성곱 신경망의 경우, 싱글모달의 컨볼루션 분기 2개를 병렬적으로 연결하여 구현하였다.

Figure 4.6은 근해 저류층에서의 쿼드모달(quad-modal) 합성곱 신경망의 구조를 나타낸다. 근해 저류층에서 사용된 합성곱 신경망에는 풀링과 제로패딩을 사용하지 않고 컨볼루션 과정을 통해서만 입력값의 압축을 진행하였다. 유체투과도, 공극률, 압력, 오일 포화도가 각각 컨볼루션 분기에 입력되고 연결 단계에서 합쳐진 후 완전연결층의 연산이 진행된다. 쿼드모달 신경망의 경우 컨볼루션 분기의 구조가 싱글모달 신경망과 다르다. 이것은 메모리 소모가 큰 컨볼루션 단계에서 학습가능 매개변수의 수를 줄임으로써 학습의 효율을 높이기 위함이다(Kim et al., 2015).

Figure 4.7은 채널 저류층에서의 쿼드모달 합성곱 신경망의 구조를 나타낸다. 근해 저류층에서와는 달리 제로패딩을 적용한 컨볼루션을 사용했기 때문에 컨볼루션층의 입력값과 출력값의 크기는 동일하다. 입력값의 압축은 두 번의 풀링층을 사용하여 진행하였다. 이러한 풀링 및 제로패딩의 사용 여부는 초매개변수의 설정 과정에 해당하며, 이 연구에서는 채널 저류층에 존재하는

불연속적인 경계면의 도출을 위해 풀링층을 사용하였다. 다중모달 합성곱 신경망에 대한 보다 자세한 내용은 케라스 코드를 첨부한 부록 B를 참조할 수 있다.



(a) (b)

Figure 4.5 (a) Structure of the ANN and (b) structure of the single-modal CNN used for the SPE10 shoreface reservoir.

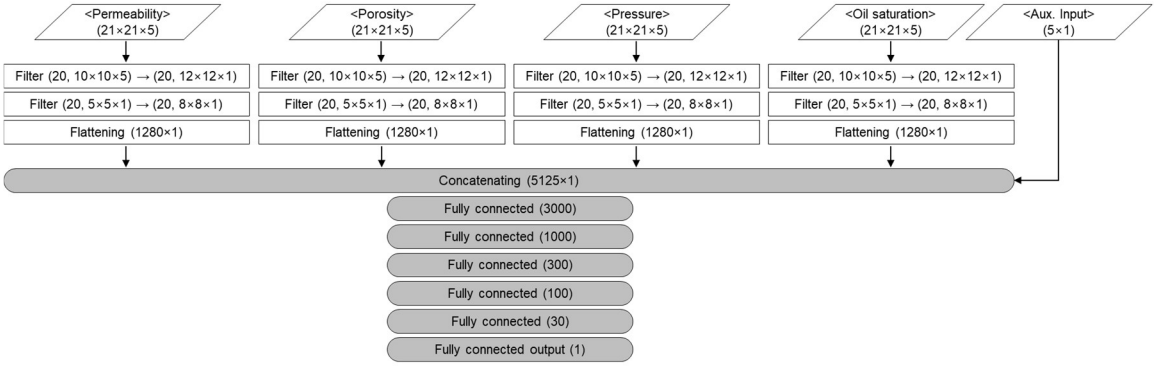


Figure 4.6 Structure of the quad-modal CNN used for the SPE10 shoreface reservoir.

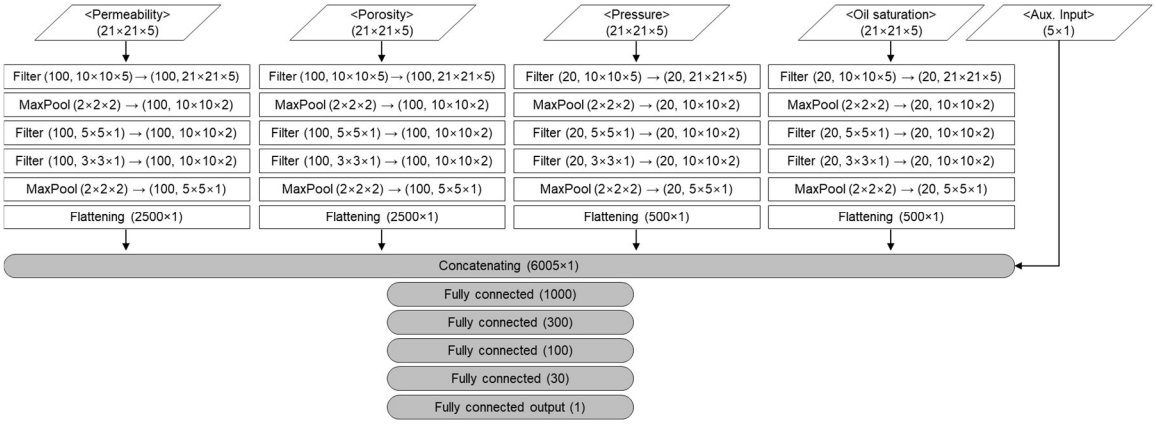


Figure 4.7 Structure of the quad-modal CNN used for the SPE10 channelized reservoir.

Table 4.3은 이 연구에서 사용한 신경망 모델의 주요 초매개변수를 나타낸다. 4–6개의 완전연결층을 사용하며, 완전연결층의 노드의 수는 최대 3,000개이다. 각각의 컨볼루션 분기에서 2–3개의 컨볼루션층을 사용하며, 컨볼루션 필터는 20–100개를 사용한다. 모든 완전연결층에는 드롭아웃이 적용되고 드롭아웃 비율은 0.2이다. Linear 함수를 사용하는 마지막 완전연결층을 제외한 모든 층에서 ReLU 함수를 활성화 함수로 사용한다. 학습률은 0.01이며, 개별 모델의 학습 횟수는 1,000회씩이다. 손실 함수는 식 (8)의 평균 제곱 오차(mean squared error, MSE)를 사용하였다. 손실 함수의 최적화 기법으로는 Adam을 사용하였다(Kingma and Ba, 2014).

$$J_1 = \mathrm{MSE} = \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} \left(\mathrm{Y}_{N^{fc}} - \widehat{\mathrm{Y}} \right)_i^2, \tag{8}$$

이 때 N_{train}은 학습 자료의 수를 의미한다.

Table 4.3 Hyper-parameters used for the SPE10 case study

Hyper-parameters	Value
Number of fully connected layers	4–6
Maximum number of nodes	3,000
Number of convolutional layers	2–3
Number of convolutional filters	20–100
Dropout rate	0.2
Activation function	ReLU
Learning rate	0.01
Number of epochs	1,000
Type of loss function	MSE
Optimizer of loss function	Adam

4.4. 사례 1: 근해 저류층

Table 4.4는 근해 저류층의 기본 정보와 추가정을 시추하는 생산 10년 시점의 누적 생산량을 나타낸다. 10년 동안 초기 부존 오일의 약 23%를 회수하였으며, 누적 물 생산량 자체는 높지 않지만 물생산비는 약 26%까지 증가한 상태이다.

Figure 4.8은 10년 동안 기존 생산정에서 생산이 진행된 시점에서의 근해 저류층의 두 번째 층의 물성 분포를 나타낸다. 하부 대수층에서의 물 유입이 진행됨에 따라 하부층(4, 5번째 층)은 전체적으로 물 포화도가 매우 높고, 상부층(첫 번째 층)에는 물 유입이 거의 되지 않았기 때문에 오일 포화도의 변화 양상을 더 뚜렷하게 보이는 두 번째 층의 물성 분포를 나타내었다. Figure 4.8(a)는 공극률 분포이며, Figure 4.8(b)와 Figure 4.8(c)는 각각 압력과 오일 포화도의 분포이다. 압력과 오일 포화도의 분포를 비교하면, 전파 속도에 따라 동적 물성의 분포 양상이 전혀 달라짐을 확인할 수 있다.

Table 4.4 Reservoir properties of the SPE10 shoreface reservoir

Reservoir properties	Value
OIIP (STB)	43,874,000
WIIP (STB)	11,049,000
Total pore volume (rbbl)	53,402,000
Cum. oil production @ 10 years (STB)	10,180,400
Cum. water production @ 10 years (STB)	648,986
Watercut @ 10 years (%)	26.13

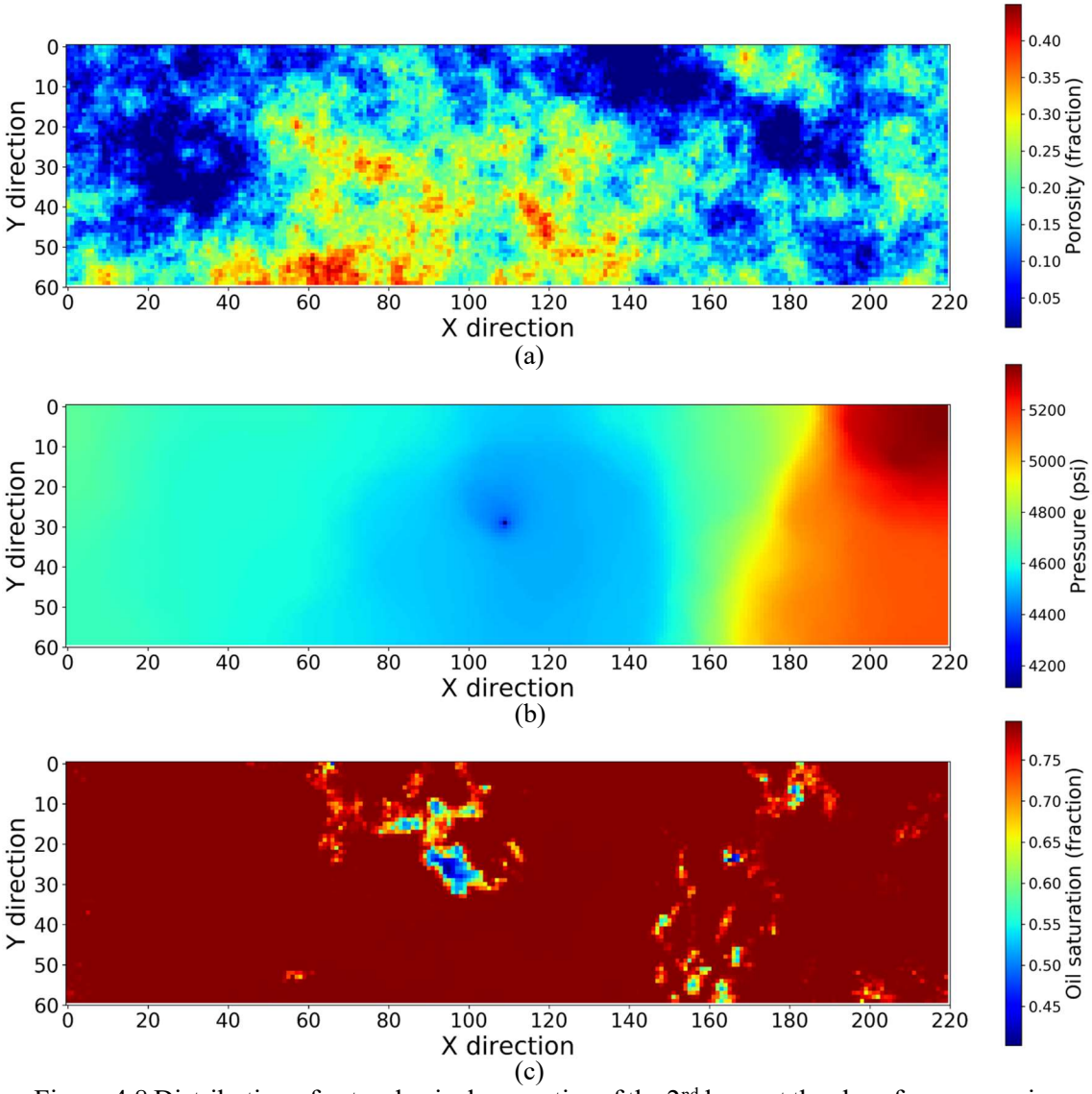


Figure 4.8 Distribution of petrophysical properties of the 2nd layer at the shoreface reservoir after ten-year of oil production from the existing well: (a) porosity, (b) pressure, and (c) oil saturation.

4.4.1. 신경망의 입력 자료로 유체투과도를 사용한 사례

이 절에서는 기존의 연구와 유사하게 유체투과도만을 입력 자료로 사용하여 인공신경망과 합성곱 신경망을 1,000회씩 학습한 결과를 나타낸다. 단, 추가적인 거리 정보는 모든 사례에서 공통적으로 학습에 사용한다.

Figure 4.9는 21×21×5 크기의 입력값을 사용하여 인공신경망과 합성곱 신경망을 학습한 결과이다. 각 그림의 x축은 저류층 시뮬레이션을 통해 획득한 참값을, y축은 프록시 모델의 출력값을 표시한다. 파란색 실선은 참조선이며, 빨간색 원은 각각의 학습 자료를 나타내는 것으로 빨간색 원이 파란색 참조선에 가깝게 위치할수록 모델의 성능이 우수하다고 할 수 있다. 또한, 파란색 글씨로 나타낸 SQRT Loss와 결정계수 R^2값으로 모델의 성능을 정량적으로 평가할 수 있다. SQRT Loss는 식 (9)와 같이 계산되며, 학습이 종료된 후 최종적으로 계산된 평균 제곱근 오차(root mean squared error, RMSE)이다. 평균 제곱근 오차 J_2는 평균 제곱 오차 J_1에 루트를 씌운 값으로 두 값의 의미는 동일하다.

$$J_2 = \text{SQRT Loss} = \text{RMSE} = \sqrt{\frac{1}{N_{train}}\sum_{i=1}^{N_{train}}\left(\mathrm{Y}_{Nfc} - \hat{\mathrm{Y}}\right)_i^2}. \tag{9}$$

신경망 모델의 구현에 사용한 케라스 프로그램에서는 기본적으로 평균 제곱 오차만을 제공하기 때문에 모델의 학습에서는 평균 제곱 오차를 사용하였다. 그러나 평균 제곱 오차(J_1)의 수치가 너무 크기 때문에 직관적인 이해를 위하여

학습이 종료된 후에 모델의 성능 평가에서는 평균 제곱근 오차(J_2)를 사용하였다. 따라서 단위는 출력값과 동일한 STB(stock tank barrel)이다. 결정계수 R^2는 자료의 참값과 출력값 사이의 상관관계를 나타내며, 값이 1에 가까울수록 강한 상관관계, 즉 모델의 성능이 우수함을 의미한다. 종합하면 SQRT Loss는 낮을수록, R^2는 높을수록 모델의 성능이 우수하다. Figure 4.9(a)는 인공신경망의 학습 세트에 대한 학습 성능을, Figure 4.9(b)는 시험 세트에 대한 시험 성능을 나타낸다. 검증 세트에 대한 성능은 시험 성능과 경향이 유사하기 때문에 생략하였다. 인공신경망의 학습 성능은 평균 제곱근 오차가 0.385 MMSTB, 결정계수가 0.868로 상당히 우수하였다. 시험 성능은 학습 성능보다 약간 떨어지는 0.525 MMSTB와 0.800을 보였으나, 일반적으로 학습 성능에 비하여 시험 성능이 낮음을 고려할 때 양호한 수치이다. 단, 생산성이 낮은 다수의 케이스들을 구분하지 못하였기 때문에 1 MMSTB 부근에 빨간 점들이 모여있는 문제가 발생하였다.

Figure 4.9(c)와 Figure 4.9(d)는 유체투과도만을 입력 자료로 사용해서 싱글모달 합성곱 신경망을 학습한 학습 성능과 시험 성능을 나타낸다. 평균 제곱근 오차의 경우 학습 성능과 시험 성능이 인공신경망에 비하여 40-50% 가량 우수한 0.3 MMSTB 내외의 값을 보였다. 결정계수 역시 상당히 개선되어 학습 성능과 시험 성능 모두 0.92 이상의 높은 값을 보였다. 단, 좌측 하단에서 보이듯 합성곱 신경망에서도 강도는 다소 약하지만 생산성이 낮은 케이스들을 구분하지 못하는 문제가 나타났다.

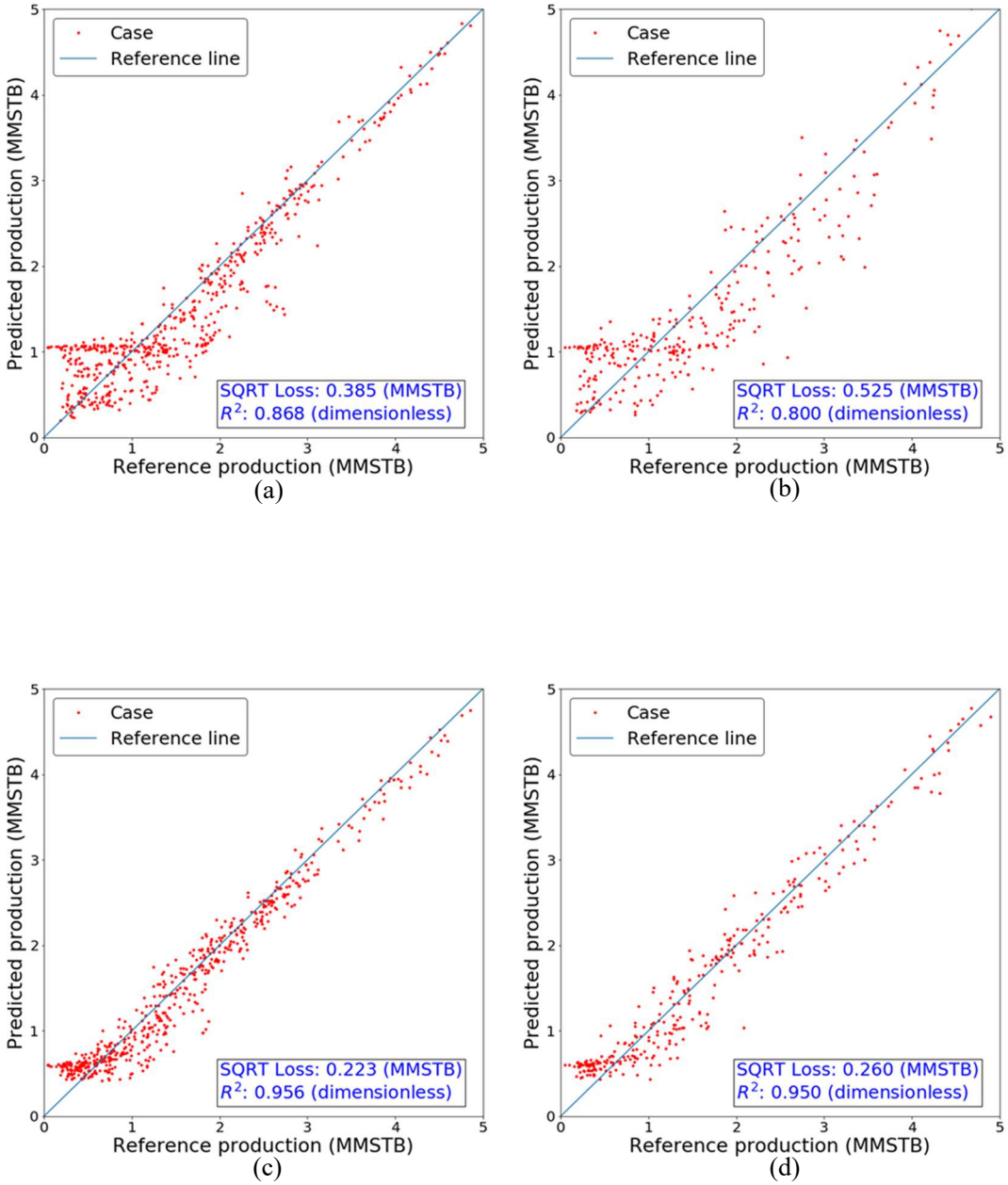


Figure 4.9 Scatter-plot to compare full-physics simulation and neural network proxy results at the SPE10 shoreface reservoir (input: permeability): (a) ANN training set, (b) ANN test set, (c) CNN training set, and (d) CNN test set.

4.4.2. 신경망의 입력 자료로 유체투과도와 포화도를 사용한 사례

유체투과도 외에 탄성파 탐사 등을 통해 오일 포화도와 같은 동적 정보를 획득한 경우를 가정하여 학습을 진행하였다. Figure 4.10(a)와 Figure 4.10(b)는 21×21×5 크기의 유체투과도와 오일 포화도 입력 배열을 사용하여 인공신경망을 학습한 결과이다. 유체투과도만을 학습한 Figure 4.9(a)와 Figure 4.9(b)에 비하여 전반적으로 학습 성능과 시험 성능이 개선되었다.

Figure 4.10(c)와 Figure 4.10(d)는 유체투과도와 오일 포화도로 듀얼모달 합성곱 신경망을 학습한 학습 성능과 시험 성능을 나타낸다. 인공신경망의 사례와 마찬가지로 유체투과도만을 학습한 Figure 4.9(c)와 Figure 4.9(d)에 비해 전반적으로 성능이 향상되었다. 학습 성능의 경우 인공신경망과 합성곱 신경망의 성능이 동일한 수준이었으며, 시험 성능은 인공신경망에 비하여 합성곱 신경망의 평균 제곱근 오차가 약 35% 우수하였고 결정계수는 4.6%p만큼 우수하였다. 생산성이 낮은 부분에서 케이스를 구분하지 못하는 현상은 두 기법 모두에서 개선되었다. 이로 미루어 서로 다른 종류의 입력값을 학습함으로써 서로를 보완할 수 있음을 확인하였다.

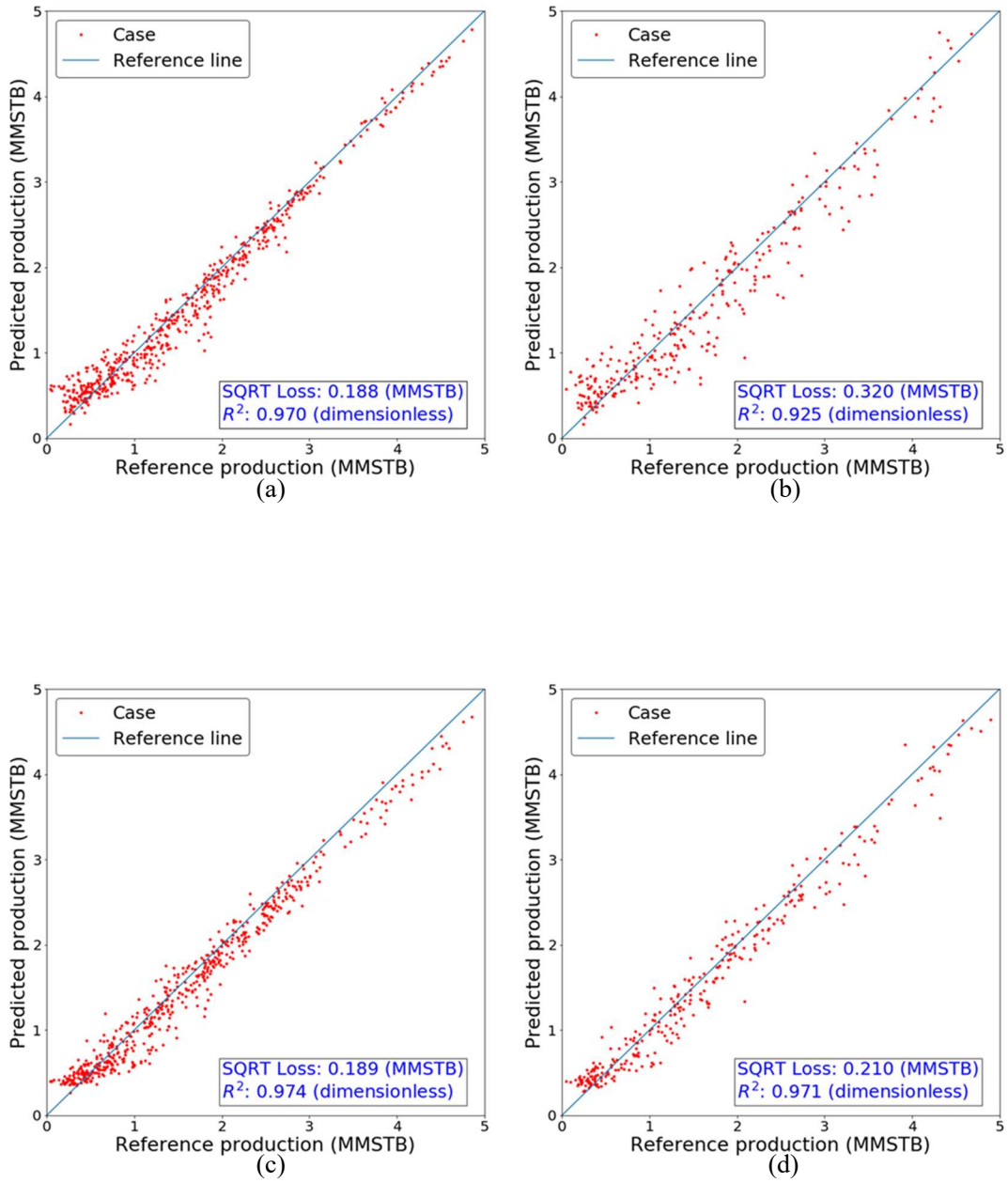


Figure 4.10 Scatter-plot to compare full-physics simulation and neural network proxy results at the SPE10 shoreface reservoir (input: permeability + oil saturation): (a) ANN training set, (b) ANN test set, (c) CNN training set, and (d) CNN test set.

4.4.3. 신경망의 입력 자료로 유체투과도, 공극률, 압력, 포화도를 사용한 사례

유체투과도와 오일 포화도 외에 공극률과 압력까지도 입력 자료로 사용할 수 있는 경우를 가정하여 학습을 진행하였다. Figure 4.11(a)와 Figure 4.11(b)는 네 종류의 입력값을 사용해 인공신경망을 학습한 경우에 있어서 입력 배열의 크기를 1×1×5로 설정한 경우의 학습 성능과 시험 성능을 나타낸다. 이러한 학습 사례는 기존의 인공신경망을 사용한 연구의 결과를 의미한다. 기존의 연구는 기울기 소실, 과적합, 부족한 연산 속도 등의 문제로 인하여 유정이 위치한 격자의 물성만을 사용하여 신경망을 학습시켰다. 현재에는 ReLU 함수, 드롭아웃, GPU 연산 등의 방법을 이용하여 상기의 문제들을 해결함으로써 더 큰 차원의 입력값을 학습할 수 있다.

Figure 4.11(c)와 Figure 4.11(d)는 21×21×5 크기의 입력 배열을 학습한 인공신경망의 학습 성능과 시험 성능을 나타낸다. Figure 4.11(a)와 Figure 4.11(b)에 비해 훨씬 우수한 성능을 보이고 있음을 확인할 수 있다. 즉, 입력 배열의 크기를 증가시켜 유정이 위치한 격자뿐 아니라 유정 주위의 물성 분포까지 학습함으로써 매우 높은 수준의 예측 모델을 획득할 수 있었다. Figure 4.11(e)와 Figure 4.11(f)는 21×21×5 크기의 입력 배열을 학습한 쿼드모달 합성곱 신경망의 학습 성능과 시험 성능을 나타낸다. 합성곱 신경망의 성능은 0.170 MMSTB 미만의 평균 제곱근 오차와 0.98 이상의 결정계수를 보이며 매우 우수하였다. 또한, 인공신경망의 사례와 마찬가지로 입력값의 종류가 늘어나자

생산성이 낮은 부분에서 발견되던 각각의 케이스를 구분하지 못하는 문제가 해결되었다. 단, 두 기법 모두 생산성이 높은 부분에서 빨간 원이 파란색의 참조선 아래에 위치하는, 즉 생산성을 과소예측하는 경향을 보였다. 그러나 이러한 경향은 수치적으로는 매우 미미할 것으로 보인다.

두 종류의 입력값을 학습한 4.4.2 절의 결과와 마찬가지로 인공신경망을 학습한 결과에서는 학습 성능과 시험 성능 간에 차이가 다소 발생하였다. 학습된 자료에 대한 학습 성능에 비해 학습되지 않은 자료에 대한 시험 성능이 낮은 것은 당연하다. 그러나 합성곱 신경망의 경우 두 성능 지표 사이의 차이(오차 0.034 MMSTB, 결정계수 0.7%p)가 크지 않지만, 인공신경망의 경우에는 그 차이(오차 0.082 MMSTB, 결정계수 1.4%p)가 상대적으로 크다. 인공신경망의 시험 성능의 절대적인 수치 자체가 매우 뛰어나기 때문에 이러한 현상을 과적합이 발생했기 때문이라고 판단하기는 어렵지만, 이러한 현상이 발생하는 근본적인 이유는 인공신경망이 구조적으로 과적합에 취약하기 때문인 것으로 보인다. 인공신경망은 학습된 자료에 대한 재현성이 합성곱 신경망보다 우수하지만 공간적 분포 특징을 감지하는 기능이 없기 때문에 더 많은 학습 자료가 요구되기 때문이다.

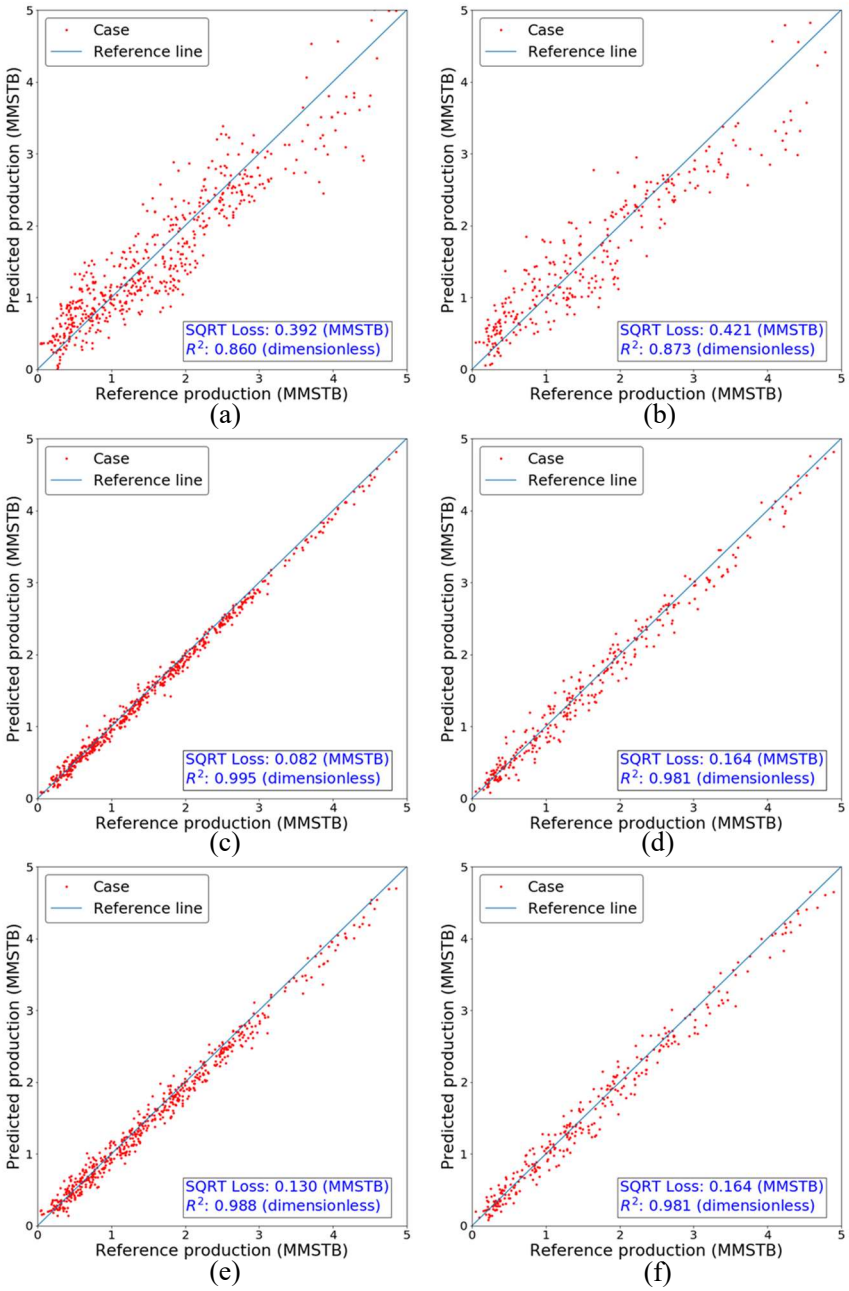


Figure 4.11 Scatter-plot to compare full-physics simulation and neural network proxy results at the SPE10 shoreface reservoir (input: permeability + porosity + pressure oil saturation): (a) ANN training set (1×1×5), (b) ANN test set (1×1×5), (c) ANN training set (21×21×5), (d) ANN test set (21×21×5), (e) CNN training set (21×21×5), and (f) CNN test set (21×21×5).

4.4.4. 신경망의 입력 자료로 유효 유체투과도를 사용한 사례

물 포화도의 함수로 나타낼 수 있는 오일의 유효 유체투과도를 신경망의 입력 자료로 사용하여 학습을 진행하였다. 오일의 유효 유체투과도는 식 (10)과 같다.

$$k_o(S_w) = kk_{ro}(S_w), \tag{10}$$

이 때, k_o, k, k_{ro}는 각각 오일의 유효 유체투과도, 절대 유체투과도, 상대 유체투과도를 의미한다. 즉, 유효 유체투과도와 상대 유체투과도 두 물성은 모두 물 포화도 S_w의 함수이다. 물 포화도의 획득 시점은 추가정이 시추되는 생산 시작 후 10년이 지난 시점으로 학습에 사용되는 오일 포화도의 획득 시점과 동일하다.

Figure 4.12(a)와 Figure 4.12(b)는 오일의 유효 유체투과도를 사용해 인공신경망을 학습한 결과를 나타낸다. 유체투과도를 학습한 4.4.1절의 결과와 비교하여 평균 제곱근 오차는 약 25% 감소한 0.288 MMSTB, 결정계수는 5.8%p 상승한 0.926의 학습 성능을 보였다. 시험 성능의 경우, 오차는 약 18% 가량 감소하여 0.430 MMSTB를, 결정계수는 6.6%p 상승하여 0.866을 보임으로써 성능이 개선되었다. 그러나 학습 성능과 시험 성능 모두 4.4.2절의 유체투과도와 오일 포화도를 동시에 학습한 사례에 비해서는 낮은 성능을 보였다.

Figure 4.12(c)와 Figure 4.12(d)는 오일의 유효 유체투과도를 사용해 싱글모달 합성곱 신경망을 학습한 결과를 나타낸다. 학습 성능에 있어서는 인공신경망과 거의 동일한 수준의 성능을 보였지만, 시험 성능에 있어서는 오차가 약 32% 개선되었고 결정계수가 7.1%p 향상되며 확연히 우수한 성능(오차 0.291 MMSTB, 결정계수 0.937)을 보였다. 4.4.1절의 유체투과도를 학습한 합성곱 신경망 사례와 비교했을 때 학습 및 시험 성능은 다소 개선되었다. 4.4.2절의 유체투과도와 오일 포화도를 함께 학습한 사례에 비해서는 유사한 수준의 성능을 보였다.

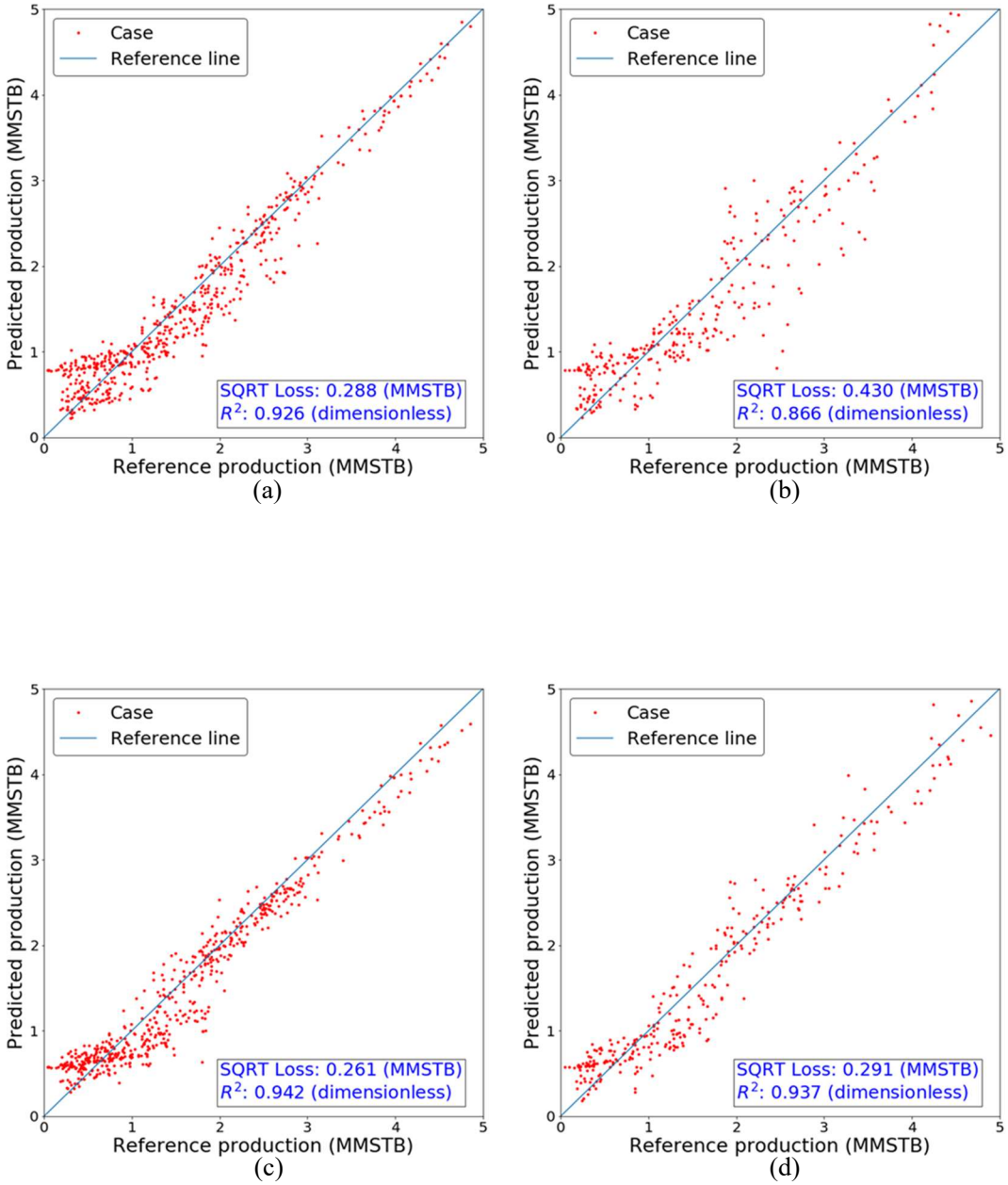


Figure 4.12 Scatter-plot to compare full-physics simulation and neural network proxy results at the SPE10 shoreface reservoir (input: effective permeability): (a) ANN training set, (b) ANN test set, (c) CNN training set, and (d) CNN test set.

4.4.5. 신경망의 학습 결과 요약

Table 4.5는 근해 저류층 모델의 유체투과도(k), 공극률(ϕ), 압력(P), 오일 포화도(S_O)를 다양하게 조합하여 인공신경망과 합성곱 신경망을 학습한 후 각각의 사례에 대한 시험 성능을 나타낸다. 모든 사례에 거리 정보는 포함된다. 각 사례는 1,000회의 학습을 5번씩 반복하여 평균을 구하였다. 입력 물성이 하나일 때에는 합성곱 신경망의 성능이 인공신경망보다 우수하다. 단, 공극률만을 학습한 경우의 시험 성능은 인공신경망과 합성곱 신경망이 거의 동일하다. 입력 물성이 늘어날수록 두 기법 모두 시험 성능이 향상되고 있으며, 두 기법 사이의 성능의 차이가 줄어드는 경향을 보인다. 유체투과도를 포함하는 듀얼모달 사례에서는 합성곱 신경망의 성능이 우수하지만, 공극귤을 포함하는 사례에서는 인공신경망의 성능이 합성곱 신경망보다 우수하다. 네 가지의 입력값을 모두 사용한 경우에는 평균 제곱근 오차는 0.200 MMSTB 미만, 결정계수는 0.75 이상으로 두 기법 모두 매우 뛰어난 성능을 보인다. 유효 유체투과도를 학습한 경우, 인공신경망과 합성곱 신경망 모두에서 유체투과도를 학습한 경우에 비하여 성능이 개선되었다.

전체적으로 공극률이 학습에 포함되는 경우 두 기법 모두 높은 성능을 보였으며, 특히 인공신경망의 성능이 매우 향상되었다. 따라서 SPE10의 근해 저류층 사례에서는 공극률의 학습 효율이 가장 뛰어나다고 판단할 수 있다. 이것은 로그 스케일의 유체투과도와 공극률 사이의 상관계수(correlation

coefficient)가 0.78로 상당히 높기 때문에 공극률이 유체의 저장 능력과 유동 능력을 동시에 반영할 수 있기 때문으로 보인다. 더욱이 Table 4.1에서 나타냈듯이 SPE10 저류층의 공극률의 표준편차는 0.086(채널 0.093)으로 일반적인 저류층 모델보다 상당히 큰 편이다(CMG, 2017). 이러한 이유로 인하여 해당 사례에서는 공극률이 유정의 생산성에 가장 큰 영향을 미치는 것으로 해석된다. 또한, 합성곱 신경망의 컨볼루션층과 풀링층의 중간 출력값을 시각화하여 부록 C로 첨부하였다.

Table 4.5 Comparison of ANN and CNN performances for the test set at the SPE10 shoreface reservoir

Input	SQRT Loss (MMSTB)		R^2 (dimensionless)	
	ANN	CNN	ANN	CNN
k (permeability)	0.532	0.320	0.795	0.922
ϕ (porosity)	0.249	0.258	0.956	0.952
p (pressure)	0.711	0.596	0.633	0.742
S_o (oil saturation)	0.854	0.784	0.473	0.555
k_o (effective oil permeability)	0.427	0.290	0.867	0.938
$k + p$	0.339	0.317	0.916	0.934
$k + S_o$	0.330	0.308	0.920	0.933
$\phi + p$	0.197	0.314	0.972	0.957
$\phi + S_o$	0.203	0.264	0.971	0.963
$k + \phi + p + S_o$	0.164	0.195	0.981	0.976

4.4.6. 신경망의 예측 결과 요약

Figure 4.13은 모든 좌표에 대해 저류층 시뮬레이션을 수행하여 획득한 근해 저류층의 좌표별 생산성을 나타낸다. 차이는 있지만 정적 물성의 분포와 유사하다는 것을 확인할 수 있다. 그러나 실제 현장 저류층 모델을 사용하는 경우에서 이러한 정답을 획득하는 것은 비싼 계산비용으로 인하여 현실적으로 불가능하다. 따라서 정답과 예측값 전체를 비교하여 예측값의 정확도를 평가하는 과정을 거칠 수 없으므로, 3.2절에서 기술한 바와 같이 소수의 후보 지점에 대해서 검증한 결과만으로 예측값의 적합도를 평가할 수 밖에 없다.

Figure 4.14와 Figure 4.15는 각각 학습된 인공신경망과 합성곱 신경망을 사용해 근해 저류층 전체에 대해 추가정의 좌표에 따른 생산성을 예측한 그림이다. Figure 4.14(a)와 Figure 4.15(a)는 유체투과도만으로 학습된 인공신경망과 합성곱 신경망 모델로 예측한 좌표별 생산성을 각각 나타낸다. 합성곱 신경망으로 학습한 모델의 생산성 예측은 비교적 정확한 경향을 보이는데 비하여 인공신경망으로 예측한 생산성은 과대예측하였음을 확인할 수 있다. 오일 포화도만으로 학습된 신경망으로 예측한 좌표별 생산성은 다른 그림과 상당히 다른 경향을 보이며, 오차가 매우 크다는 것을 알 수 있다. 또한, 명확한 최대 생산량 지점을 도출하지도 못했다(Figure 4.14(b)와 Figure 4.15(b)). 유체투과도와 오일 포화도를 학습한 인공신경망이 도출한 결과 역시 상당히 높은 최대 생산량 값을 도출하였다(Figure 4.14(c)). 반면에 유체투과도와 오일

포화도를 학습한 합성곱 신경망으로 예측한 생산량은 상당히 합리적인 결과를 보였다(Figure 4.15(c)). 실제로는 학습되지 않은 지점에 대해서 저류층 시뮬레이션을 수행하기 전까지는 인공신경망과 합성곱 신경망의 예측 결과의 정확성을 확인할 수는 없다. 단, Table 4.5에서 확인할 수 있듯이 신경망의 시험 성능을 통해 전체적인 예측 성능을 유추할 수 있다. 네 가지의 입력값을 모두 학습한 인공신경망과 합성곱 신경망의 시험 성능은 오차는 0.2 MMSTB 미만, 결정계수는 0.97 이상일 정도로 매우 우수하였기 때문에 저류층 시뮬레이션을 수행하기 이전 시점에서도 전체적인 예측 성능이 충분히 높을 것으로 판단할 수 있었다.

실제 현장 자료를 학습 자료로 사용하는 경우에는 자료의 불확실성이 존재하는데다 자료 사이의 질적 차이도 크다. 따라서 연구자는 자료 사이의 질적 수준을 파악하고 학습에 사용할 자료를 선별할 필요가 있다. 그러나 이 연구에서는 가상 저류층을 사용한 저류층 시뮬레이션 결과를 기반으로 하기 때문에 자료 간의 질적 차이 및 불확실성이 존재하지 않는다. 따라서 학습 성능과 시험 성능이 매우 우수하며, 가용한 모든 입력값을 학습하고 과적합 문제도 발생하지 않은 Figure 4.15(d)의 모델이 가장 뛰어나다고 판단하였다.

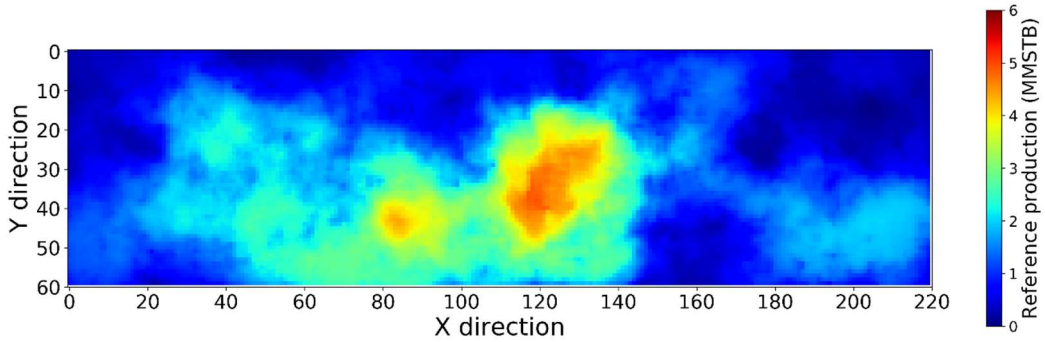


Figure 4.13 Reference productivity map of the SPE10 shoreface reservoir.

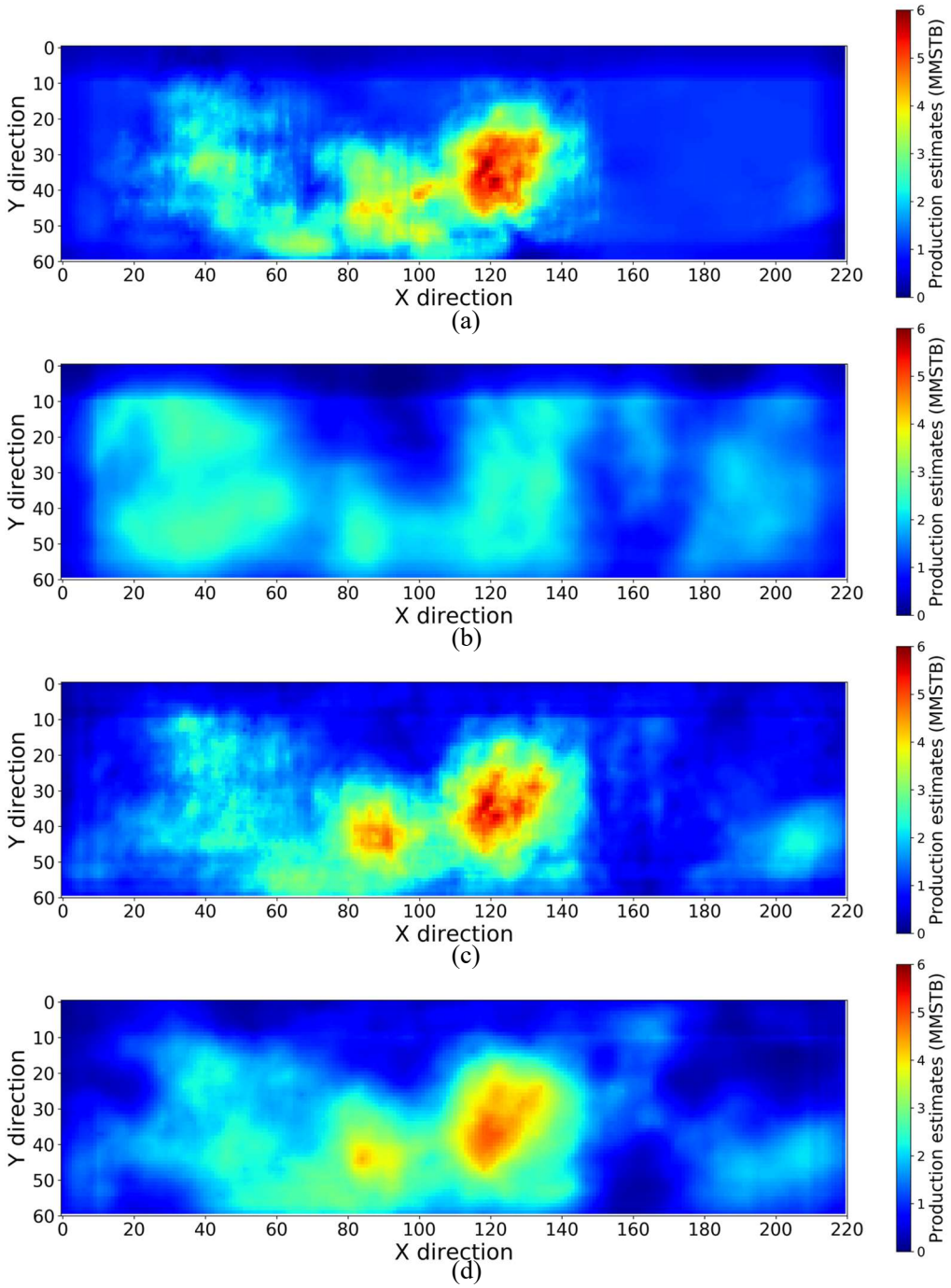


Figure 4.14 Productivity maps of the SPE10 shoreface reservoir using ANN: (a) productivity map obtained using k, (b) productivity map obtained using S_o, (c) productivity map obtained using $k + S_o$, and (d) productivity map obtained using $k + \phi + P + S_o$.

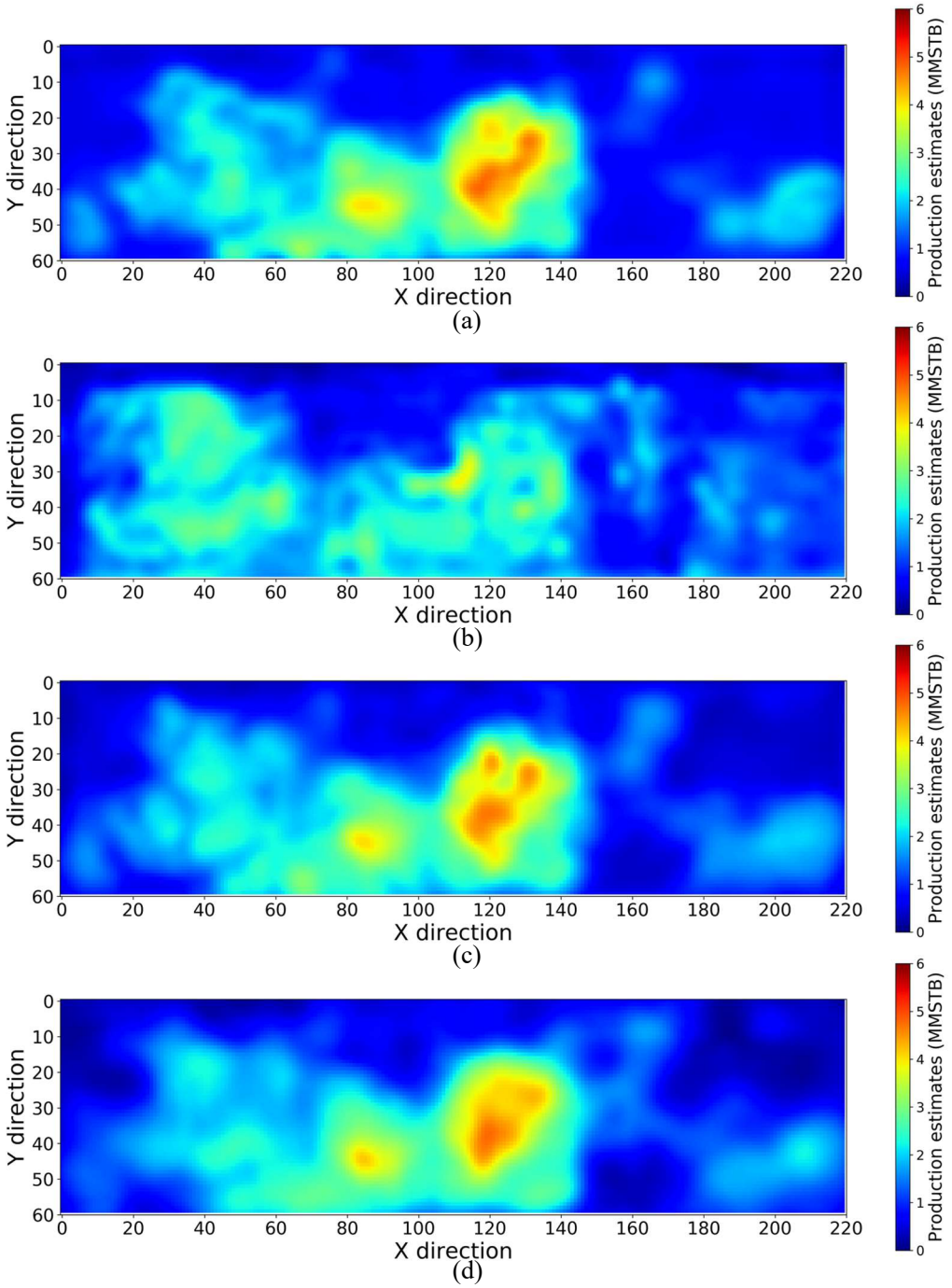


Figure 4.15 Productivity maps of the SPE10 shoreface reservoir using CNN: (a) productivity map obtained using k, (b) productivity map obtained using S_o, (c) productivity map obtained using $k + S_o$, and (d) productivity map obtained using $k + \phi + P + S_o$.

Table 4.6은 Figure 4.15(d)의 모델을 사용해 예측한 가장 높은 생산성을 가지는 지점 상위 20곳에 대해 실제 저류층 시뮬레이션을 수행한 결과와 상대 오차를 나타낸다. 상대 오차 ε은 식 (11)과 같이 계산한다.

$$\varepsilon = \frac{|d^{NN} - d^{sim}|}{d^{sim}} \times 100 \ \ (\%), \tag{11}$$

이 때 d^{NN}은 신경망 모델로 예측한 생산량이며, d^{sim}은 저류층 시뮬레이션으로 예측한 생산량이다. 합성곱 신경망의 예측값은 미세하게 과소예측하는 경향을 보이지만 20개 시나리오의 평균 상대 오차가 1.5%에 불과할 정도로 정확한 결과를 보였다. 합성곱 신경망이 도출한 최적의 위치 (120, 39) 지점은 상대 오차가 2.95%였으며, 실제 저류층 시뮬레이션에서도 20개 시나리오 중 최대의 생산량을 보였다. 20개 시나리오 중 상대 오차가 가장 큰 경우조차 3.5%에 불과하였다. 저류층 시뮬레이션의 검증까지 수행한 결과 추가정의 최적 위치는 1번 사례의 (120, 39)이다. 그러나 20개 시나리오에 대한 신경망 모델의 예측값과 시뮬레이션 결과값 사이의 결정계수는 0.64로 앞서의 시험 세트에 대한 결정계수와 비교해서 다소 낮았다. 이러한 결과는 의사결정 과정의 최종 결정은 전적으로 프록시를 활용하기보다는 물리적 법칙에 기반한 시뮬레이션이 더욱 정확한 결과를 도출할 수 있음을 시사한다. 또한, 프록시로 선별한 지점에 대하여 추가로 수행한 시뮬레이션 결과를 후속 프록시의 학습자료에 추가함으로써 후속 프록시를 개선 및 활용하는 순차 프록시(sequential proxy) 연구의 필요성을 내포하고 있다.

Table 4.6 Comparison of simulation and CNN results for the qualified 20 infill well scenarios at the SPE10 shoreface reservoir

Rank	X location	Y location	Predicted production (MMSTB)	Reference production (MMSTB)	Relative error (%)
1	120	39	4.786	4.931	2.95
2	121	38	4.783	4.823	0.83
3	121	39	4.782	4.891	2.22
4	120	38	4.776	4.883	2.19
5	120	40	4.774	4.881	2.19
6	121	40	4.757	4.874	2.42
7	121	37	4.752	4.784	0.68
8	120	41	4.738	4.808	1.45
9	120	37	4.738	4.844	2.20
10	122	38	4.737	4.737	0.00
11	122	39	4.727	4.679	1.02
12	119	40	4.727	4.835	2.24
13	119	39	4.722	4.893	3.50
14	122	37	4.712	4.689	0.48
15	119	41	4.710	4.781	1.49
16	121	41	4.705	4.836	2.71
17	119	38	4.698	4.856	3.24
18	122	40	4.690	4.706	0.35
19	121	36	4.689	4.754	1.36
20	120	42	4.687	4.647	0.85
	Average		4.734	4.807	1.50

4.5. 사례 2: 채널 저류층

사례 2에서는 Figure 4.1(b)에 나타난 채널 저류층에서의 추가 생산정 위치 최적화 문제를 신경망으로 탐색한다. Figure 4.16은 채널 저류층의 두 번째 층의 공극률, 압력, 오일 포화도의 분포를 나타낸다. 채널 저류층의 학습 과정에서는 총 504개의 자료를 사용하였다. 추가정이 채널의 외부, 즉 유체투과도가 매우 낮은 셰일층에 위치하는 경우를 인공신경망에 다수 학습시키는 것은 시뮬레이션 수에 비례하는 학습 비용을 불필요하게 증가시킬 수 있기 때문이다. 물론 셰일층에 대해서도 학습한다면 신경망 모델의 전체 예측 성능은 향상될 수 있겠지만 이러한 성능 향상 효과보다 더 많은 학습 자료의 획득에 따른 효율성 감소 효과가 더 클 것으로 판단하였다. 504개의 학습 자료 중 104개의 자료를 시험 세트로 사용하고, 나머지 자료의 20%를 검증 세트로 사용하였기 때문에 학습 세트와 검증 세트, 시험 세트의 비율은 320:80:104이다.

Table 4.7은 채널 저류층의 기본 정보와 추가정을 시추하는 시점인 생산 10년 시점의 누적 생산량을 나타낸다. 오일의 회수량은 약 19%이며, 물생산비는 31.56%이다.

Table 4.7 Reservoir properties of the SPE10 channelized reservoir

Reservoir properties	Value
OIIP (STB)	31,104,000
WIIP (STB)	7,832,800
Total pore volume (rbbl)	37,858,000
Cum. oil production @ 10 years (STB)	5,975,770
Cum. water production @ 10 years (STB)	616,521
Watercut @ 10 years (%)	31.56

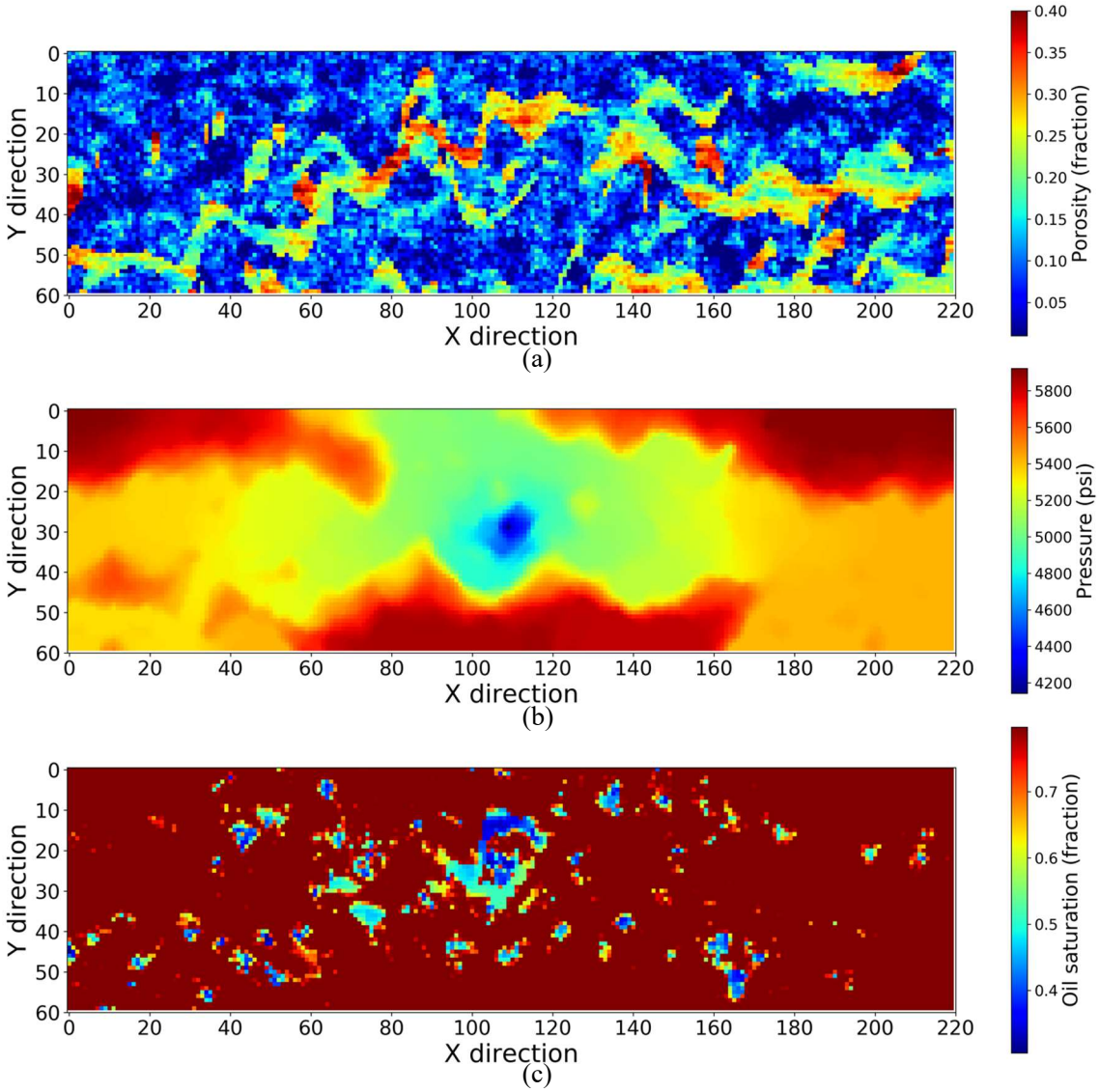


Figure 4.16 Distribution of petrophysical properties of the 2nd layer at the SPE10 channelized reservoir after ten-year of oil production from the existing well: (a) porosity, (b) pressure, and (c) oil saturation.

4.5.1. 신경망의 입력 자료로 유체투과도를 사용한 사례

이 절에서는 유체투과도만을 사용하여 인공신경망과 합성곱 신경망을 1,000회씩 학습한 결과를 나타낸다. 추가적인 거리 정보는 모든 사례에서 공통적으로 사용되었다. Figure 4.17(a)와 Figure 4.17(b)는 21×21×5 크기의 입력 배열을 학습한 인공신경망의 학습 성능과 시험 성능을 나타낸다. 인공신경망의 학습 성능은 평균 제곱근 오차 0.084 MMSTB, 결정계수 0.978로 매우 우수하다. 그러나 시험 성능의 경우 오차가 0.656 MMSTB로 크게 증가하였고 결정계수는 0.121로 사실상 연관성이 없는 수준의 결과를 보였다. 이러한 결과, 즉 학습 성능과 시험 성능간 결정계수의 큰 편차는 명백한 과적합 발생의 증거로서 채널 저류층에서는 유체투과도만으로 인공신경망을 학습하는 경우 의미있는 결과를 내는 것이 어려움을 확인하였다.

Figure 4.17(c)와 Figure 4.17(d)는 유체투과도만으로 합성곱 신경망을 학습한 결과에 대한 학습 성능과 시험 성능을 나타낸다. 합성곱 신경망의 학습 성능은 오차가 0.242 MMSTB, 결정계수가 0.785로 인공신경망의 학습 성능에 비해 낮은 수준의 성능을 보였다. 그러나 시험 성능에 있어서는 0.267 MMSTB의 오차와 0.696의 결정계수를 나타냄으로써 인공신경망보다 우수한 성능을 보임을 확인하였다. 합성곱 신경망의 학습 및 검증 성능에서 획득한 오차와 결정계수의 편차 또한 인공신경망의 결과에 비해 작았다. 이러한 결과는 인공신경망에서 발생한 과적합 문제가 합성곱 신경망에서는 크게 나타나지 않음을 의미한다.

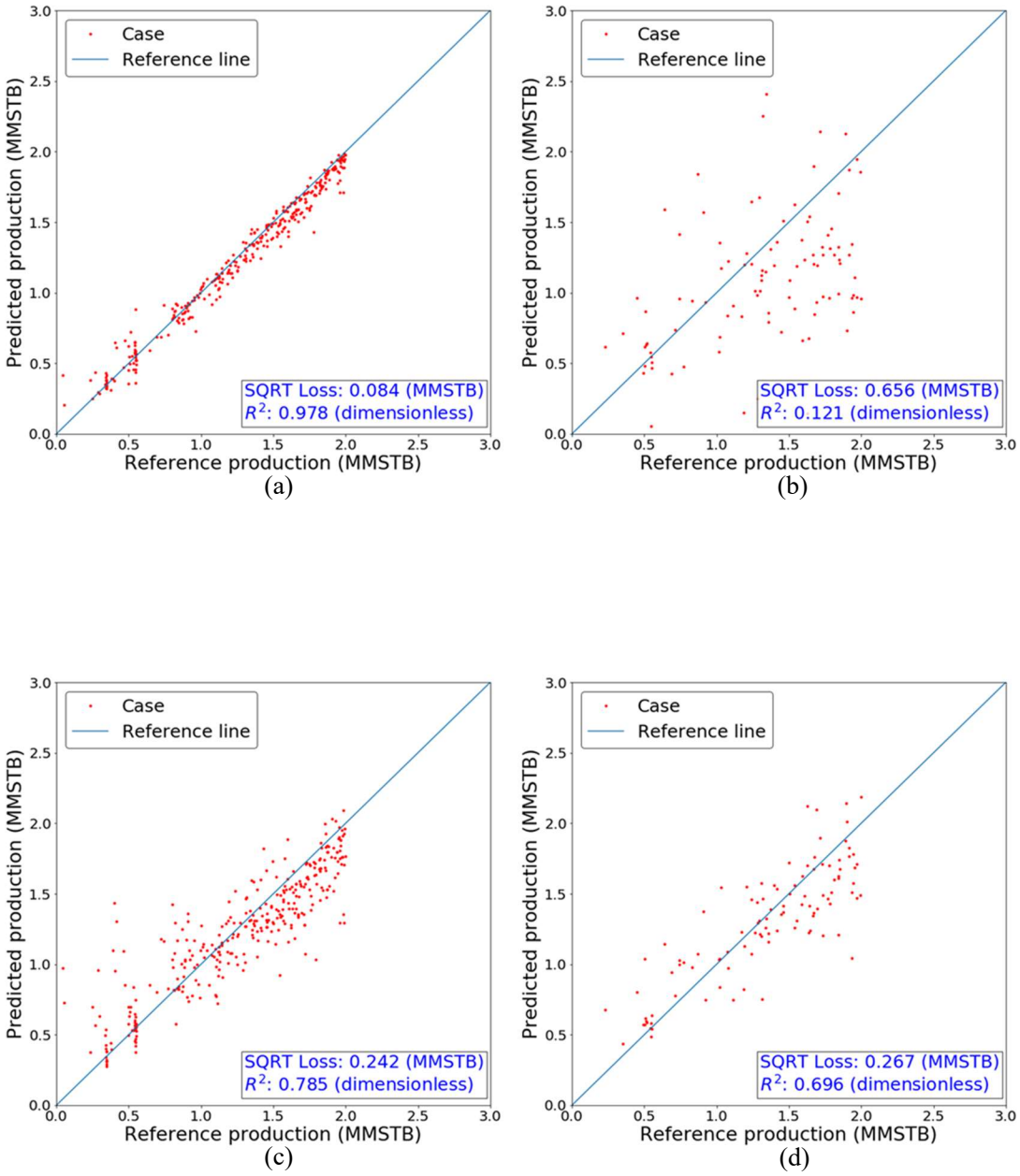


Figure 4.17 Scatter-plot to compare full-physics simulation and neural network proxy results at the SPE10 channelized reservoir (input: permeability): (a) ANN training set, (b) ANN test set, (c) CNN training set, and (d) CNN test set.

4.5.2. 신경망의 입력 자료로 유체투과도와 포화도를 사용한 사례

유체투과도 외에 탄성파 탐사를 통해 오일 포화도를 추가적으로 획득하여 학습에 사용한 상황을 가정하였다. Figure 4.18(a)와 Figure 4.18(b)는 21×21×5 크기의 입력 배열을 학습한 인공신경망의 학습 성능과 시험 성능을 나타낸다. 유체투과도와 오일 포화도를 학습한 인공신경망의 학습 성능은 개형이 직선에 가까울 정도로 매우 우수하였다. 그러나 시험 성능은 학습 성능에 비해 훨씬 좋지 않은 0.434 MMSTB의 오차와 0.425의 결정계수를 보였다. 결론적으로 오일 포화도를 추가적으로 학습했음에도 불구하고 상당한 과적합이 발생하였음을 확인할 수 있다.

Figure 4.18(c)와 Figure 4.18(d)는 유체투과도와 오일 포화도로 듀얼모달 합성곱 신경망을 학습한 결과이다. 학습 성능은 0.271 MMSTB의 오차와 0.730의 결정계수를 보이면서 인공신경망 결과에 비해 약간 감소하였다. 그러나 시험 성능에 있어서는 인공신경망 결과보다 훨씬 우수한 0.277 MMSTB의 오차와 0.677의 결정계수를 보였다. 또한, 학습 성능과 시험 성능 간의 큰 차이(오차 0.377 MMSTB, 결정계수 56.6%p)를 보인 인공신경망에 비해 합성곱 신경망은 학습 성능과 시험 성능 간의 차이(오차 0.006 MMSTB, 결정계수 5.3%p)가 크지 않았다. 입력 자료가 추가되었음에도 과적합 문제는 인공신경망에서만 두드러지게 발생하였음을 확인하였다.

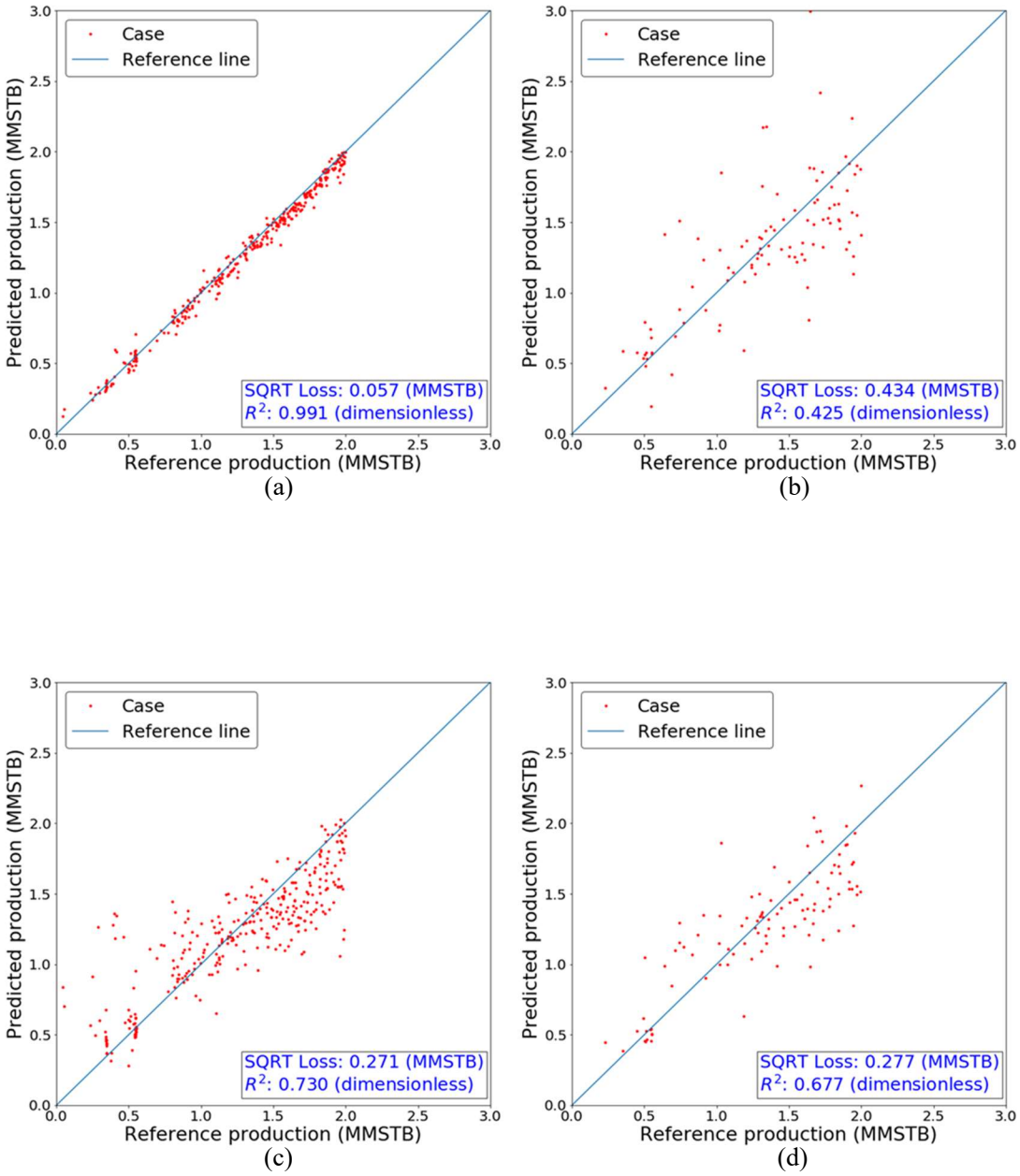


Figure 4.18 Scatter-plot to compare full-physics simulation and neural network proxy results at the SPE10 channelized reservoir (input: permeability + oil saturation): (a) ANN training set, (b) ANN test set, (c) CNN training set, and (d) CNN test set.

4.5.3. 신경망의 입력 자료로 유체투과도, 공극률, 압력, 포화도를 사용한 사례

유체투과도와 오일 포화도에 공극률과 압력을 추가하여 네 가지의 입력값으로 인공신경망과 합성곱 신경망을 1,000회씩 학습하였다. Figure 4.19(a)와 Figure 4.19(b)는 1×1×5 크기의 입력 배열을 사용한 인공신경망의 학습 성능과 시험 성능을 각각 나타낸다. Figure 4.19(c)와 Figure 4.19(d)는 21×21×5 크기의 입력 배열을 사용한 인공신경망의 학습 성능과 시험 성능을 각각 나타낸다. 입력 배열이 커지면서 학습 성능과 시험 성능이 모두 대폭적으로 개선되었음을 확인할 수 있으며, 학습 성능의 경우 0.999의 결정계수로 직선과 다름 없는 수준을 보였다. 반면에 시험 성능은 0.238 MMSTB의 오차와 0.774의 결정계수를 보임으로써, 학습 성능에 비해서는 다소 낮은 수준의 성능을 보였다.

Figure 4.19(e)와 Figure 4.19(f)는 21×21×5 크기의 입력 배열을 학습한 합성곱 신경망의 학습 성능과 시험 성능을 각각 나타낸다. 합성곱 신경망의 학습 성능은 오차 0.203 MMSTB, 결정계수 0.859로 인공신경망(Figure 4.19(c))에 비해 다소 낮았으나, 시험 성능은 평균 제곱근 오차 0.233 MMSTB, 결정계수 0.786으로 거의 동일한 수준의 성능을 보였다. 합성곱 신경망의 학습 성능과 시험 성능 간의 차이는 오차 0.030 MMSTB, 결정계수 7.3%p로 매우 양호하였다. 이러한 결과로 미루어 채널 저류층에서의 합성곱 신경망에서는 과적합 문제가 거의 발생하지 않은 것으로 보이며, 따라서 시험 세트 외의 다른 지점에 대한 예측 성능도 우수할 것으로 판단하였다.

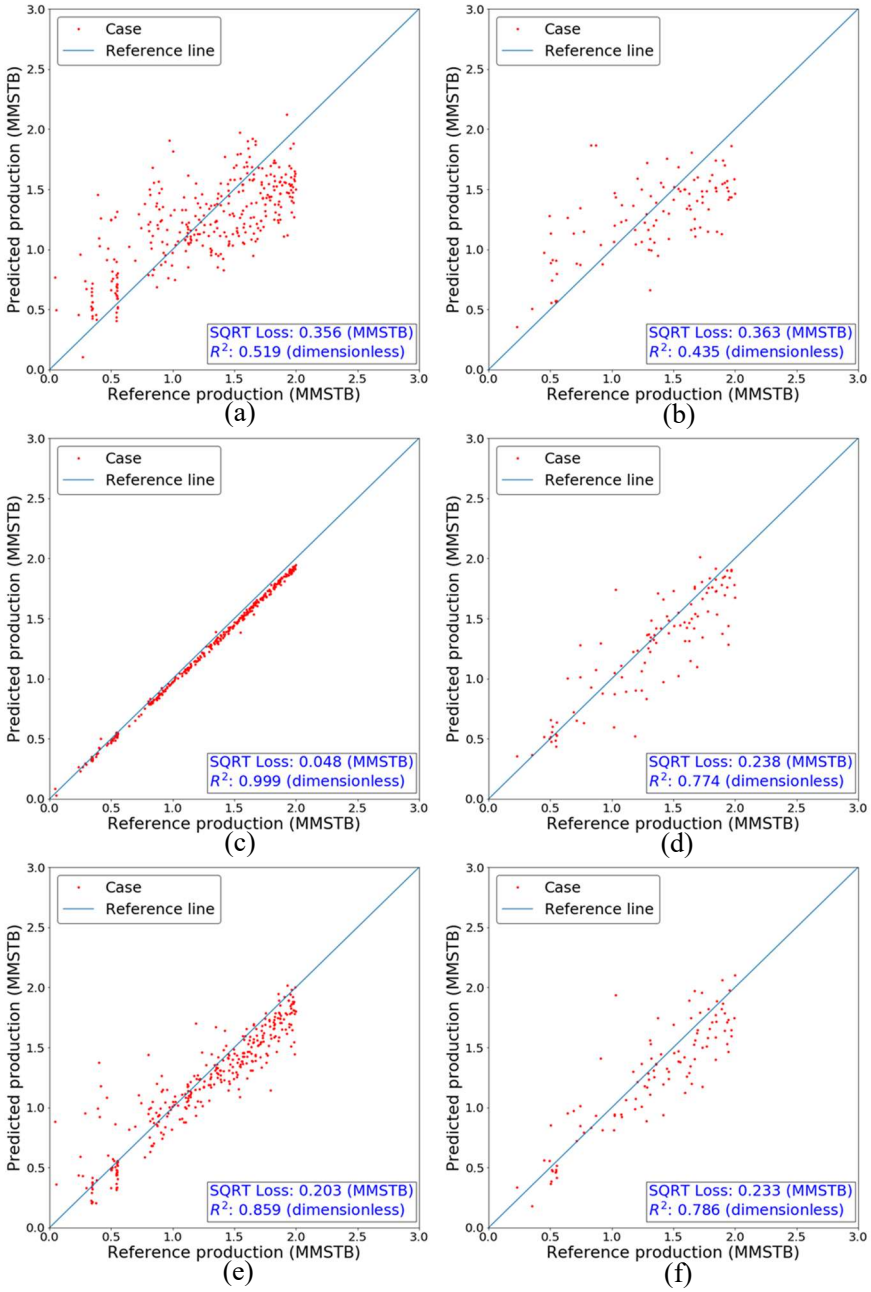


Figure 4.19 Scatter-plot to compare full-physics simulation and neural network proxy results at the SPE10 channelized reservoir (input: permeability + porosity + pressure oil saturation): (a) ANN training set (1×1×5), (b) ANN test set (1×1×5), (c) ANN training set (21×21×5), (d) ANN test set (21×21×5), (e) CNN training set (21×21×5), and (f) CNN test set (21×21×5).

4.5.4. 신경망의 입력 자료로 유효 유체투과도를 사용한 사례

오일의 유효 유체투과도를 입력 자료로 사용하여 인공신경망과 합성곱 신경망을 학습하였다. Figure 4.20(a)와 Figure 4.20(b)는 인공신경망의 학습 및 시험 성능을 나타낸다. 인공신경망의 학습 성능과 시험 성능간의 차이가 매우 컸기 때문에 과적합이 발생하였다고 판단하였다. 4.5.1절의 유체투과도를 학습한 인공신경망의 사례와 비교했을 때 학습 성능과 시험 성능이 개선 효과가 크지 않음을 확인하였다. 시험 성능의 경우 약 9.3%의 오차의 감소를 보였지만, 결정계수가 1.1%p 하락하였기 때문에 유의미한 성능 개선을 확인할 수 없었다.

Figure 4.20(c)와 Figure 4.20(d)는 합성곱 신경망의 학습 성능과 시험 성능을 각각 나타낸다. 합성곱 신경망의 학습 성능은 인공신경망보다 열악했으나 시험 성능에서는 월등히 우수한 결과를 보였다. 그러나 4.5.1절의 유체투과도를 학습한 사례와 비교했을 때에는 두드러진 성능 개선 효과가 없었다.

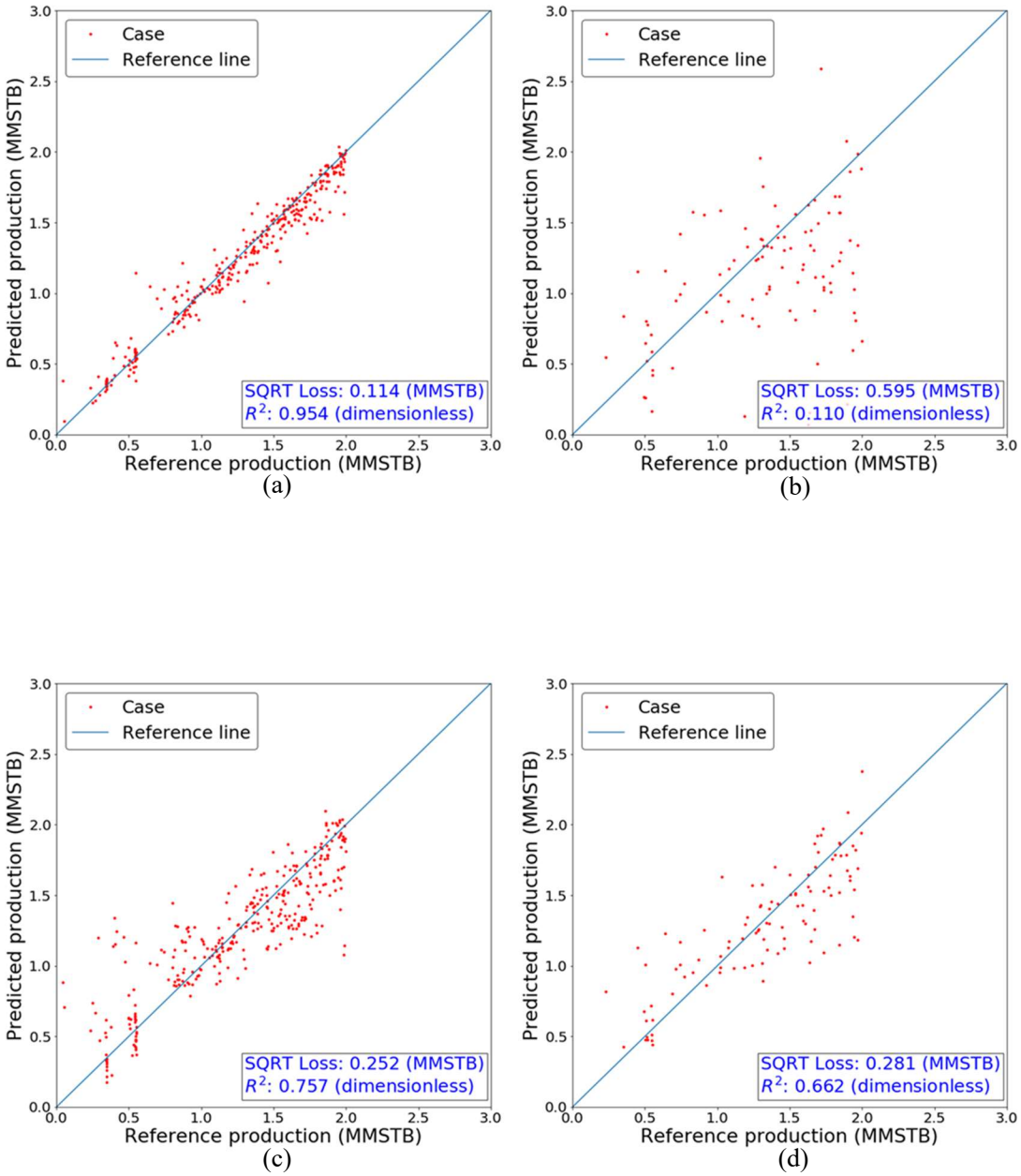


Figure 4.20 Scatter-plot to compare full-physics simulation and neural network proxy results at the SPE10 channelized reservoir (input: effective permeability): (a) ANN training set, (b) ANN test set, (c) CNN training set, and (d) CNN test set.

4.5.5. 신경망의 학습 결과 요약

Table 4.8은 채널 저류층 모델의 유체투과도, 공극률, 압력, 오일 포화도를 다양하게 조합하여 인공신경망과 합성곱 신경망을 학습한 후 계산한 시험 성능을 나타낸다. 모든 사례에 거리 정보는 포함되며, 각 사례는 1,000회의 학습을 5번씩 반복하여 구한 평균을 나타낸다. 오일 포화도만을 학습한 사례를 제외하면, 모든 사례에서 합성곱 신경망의 시험 성능이 인공신경망의 시험 성능보다 오차는 작고, 결정계수는 높은 경향을 보였다. 유일한 예외 사례(S_O)에서도 인공신경망의 수치가 다소 우수할 뿐 거의 비슷한 수준의 수치를 보였다. 위 결과는 향후 신경망 기반 유정 위치 선정 기법을 실제 저류층에 적용할 때 포화도 등 유체 함유율 뿐만아니라 유체투과도 등 유체의 유동능력에 관한 인자를 신경망의 입력값으로 고려할 필요가 있음을 의미한다.

입력값의 종류가 적은 싱글모달이나 듀얼모달 사례에서는 인공신경망과 합성곱 신경망 사이의 성능 차이가 근해 저류층 사례보다 증가하여, 합성곱 신경망이 확연히 우월한 성능을 보였다. 이러한 현상은 채널 저류층에는 샌드 채널과 같은 물성 분포에 있어서 명확한 불연속면이 존재하기 때문으로 해석하였다. 합성곱 신경망 기법은 사물 인식 분야에서 매우 뛰어난 성능을 발휘하는 기법으로, 채널의 경계를 찾는 과정은 사물 인식에서 대상의 윤곽을 찾는 과정과 유사하다고 볼 수 있기 때문이다. 단, 입력값의 종류가 많은 쿼드모달 사례에서는 두 기법의 성능 차이가 거의 발생하지 않았다. 이와 같은

결과를 통하여 비록 인공신경망은 입력값의 공간적 분포 패턴을 파악하는데 한계가 있지만, 양질의 학습 자료를 학습할 수 있다면 그러한 한계를 극복할 수 있을 것으로 판단하였다. 인공신경망은 합성곱 신경망에 비하여 학습 소요 시간이 적고, 최적화 과정이 비교적 간단하기 때문에 가용 자료가 충분하다면 높은 활용도를 보일 것으로 기대된다. 또한, 합성곱 신경망의 컨볼루션층과 풀링층의 중간 출력값을 시각화하여 부록 C로 첨부하였다.

Table 4.8 Comparison of ANN and CNN performances for the test set at the SPE10 channelized reservoir

Input	SQRT Loss (MMSTB)		R^2 (dimensionless)	
	ANN	CNN	ANN	CNN
k (permeability)	0.658	0.269	0.123	0.697
ϕ (porosity)	0.356	0.262	0.577	0.712
p (pressure)	0.670	0.304	0.042	0.618
S_o (oil saturation)	0.305	0.356	0.610	0.587
k_o (effective oil permeability)	0.591	0.287	0.113	0.659
$k + p$	0.471	0.277	0.346	0.691
$k + S_o$	0.438	0.281	0.420	0.673
$\phi + p$	0.305	0.274	0.676	0.699
$\phi + S_o$	0.249	0.243	0.736	0.754
$k + \phi + p + S_o$	0.235	0.234	0.773	0.775

Table 4.9는 채널 저류층 모델의 학습 자료를 사용해 교차 검증(cross validation)을 수행한 결과를 나타낸다. 유체투과도, 공극률, 압력, 오일 포화도의 네 종류의 입력 자료를 사용하는 쿼드모달 사례에 대해 인공신경망과 합성곱 신경망으로 교차 검증을 수행하였다. 해당 교차 검증을 위해 채널 저류층의 학습 자료를 다섯 부분으로 나눈 후, 학습 세트와 검증 세트, 시험 세트의 구성을 달리하여 학습을 수행하였다. 교차 검증 결과, 쿼드모달 사례에 있어서 인공신경망과 합성곱 신경망의 시험 성능은 거의 동일한 수준으로 나타났다.

Table 4.9 Cross validation of ANN and CNN at the SPE10 channelized reservoir
(*Avg.: average, Std.: standard deviation)

Performance	ANN			CNN		
	Training	Validation	Test	Training	Validation	Test
Avg. SQRT Loss (MMSTB)	0.043	0.301	0.296	0.177	0.307	0.301
Avg. R^2 (dimensionless)	0.998	0.660	0.661	0.879	0.643	0.645
Std. SQRT Loss (MMSTB)	0.009	0.042	0.036	0.020	0.049	0.061
Std. R^2 (dimensionless)	0.001	0.076	0.071	0.027	0.100	0.124

4.5.6. 신경망의 과적합 검증

근해 저류층과 달리 채널 저류층의 학습 결과에서는 학습 세트에 대한 성능과 시험 세트에 대한 성능의 차이가 큰 과적합이 발생한 사례가 특히 인공신경망의 학습 결과에서 많이 발견되었다. 학습 과정에서의 과적합의 발생 여부는 Figure 4.21과 같이 학습 세트의 손실은 감소하지만 검증 세트의 손실은 증가하는 그래프의 경향으로 판단할 수 있다. 해당 그림은 0부터 9까지의 숫자를 나타낸 손글씨 자료인 MNIST(Modified National Institute of Standards and Technology) 데이터셋을 분류하는 문제의 손실 함수 경향을 나타낸 것이다. 이와 같은 과적합이 발생하는 경우 모델의 범용적 성능이 감소하게 되므로 과적합이 발생하기 전에 학습을 종료할 필요가 있다. 해당 MNIST 데이터셋을 사용한 케라스 예제는 부록 D로 첨부하였다.

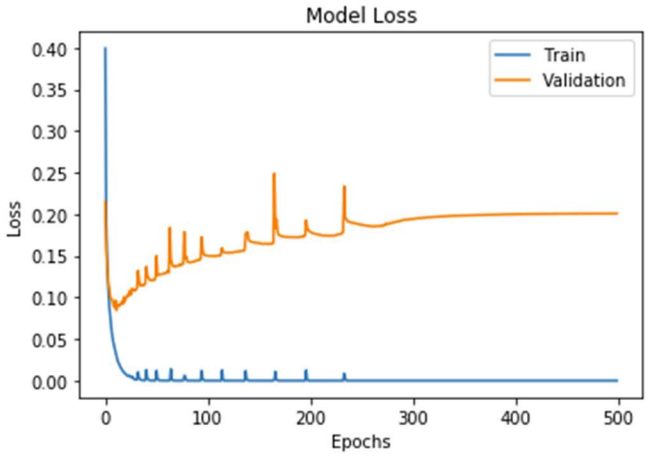


Figure 4.21 Loss function trend of MNIST classification problem.

Figure 4.22(a)와 Figure 4.22(b)는 채널 저류층에서 유체투과도를 학습한 인공신경망과 합성곱 신경망의 손실 함수 경향을 각각 나타낸다. 두 모델에는 모두 과적합을 방지하기 위하여 드롭아웃이 적용되었으나, 그럼에도 불구하고 인공신경망의 손실 함수는 학습 초기에 검증 세트에 대한 손실이 증가하면서 과적합이 발생한 현상을 보이고 있다. 반면에 합성곱 신경망에서는 두 손실 함수가 꾸준히 감소하고 있으며 증가하는 경향을 보이지 않는다.

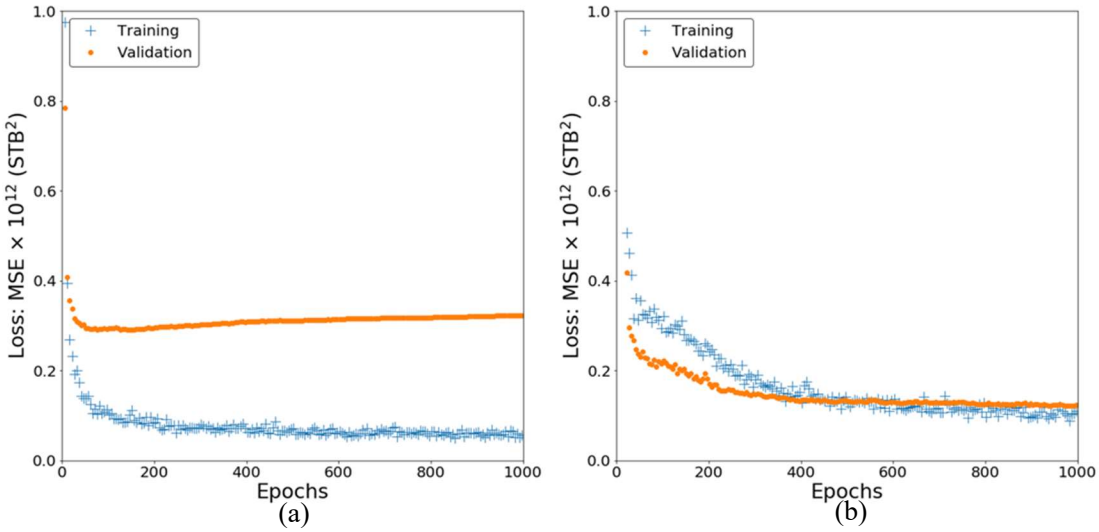


Figure 4.22 Loss function trend at the channelized reservoir (input: permeability): (a) ANN and (b) CNN.

Figure 4.23(a)와 Figure 4.23(b)는 학습 자료로 유체투과도와 공극률, 압력, 오일 포화도를 사용한 인공신경망과 합성곱 신경망의 손실 함수 경향을 각각 나타낸다. 인공신경망의 경우 하나의 입력 자료를 학습한 앞서의 사례(Figure 4.22(a))와 달리 검증 세트의 손실 함수가 증가하는 경향을 보이지 않는다. 즉, 입력 자료가 증가하여 학습할 양이 많아짐에 따라 인공신경망에서의 과적합 발생의 문제가 억제되었음을 확인할 수 있다. 또한, 합성곱 신경망에서도 과적합은 발생하지 않았다. 이상의 결과를 바탕으로 인공신경망은 과적합에 취약하기는 하지만 네 종류의 입력 자료를 사용하는 경우에는 과적합 발생에 관한 문제는 해소될 것으로 판단하였다.

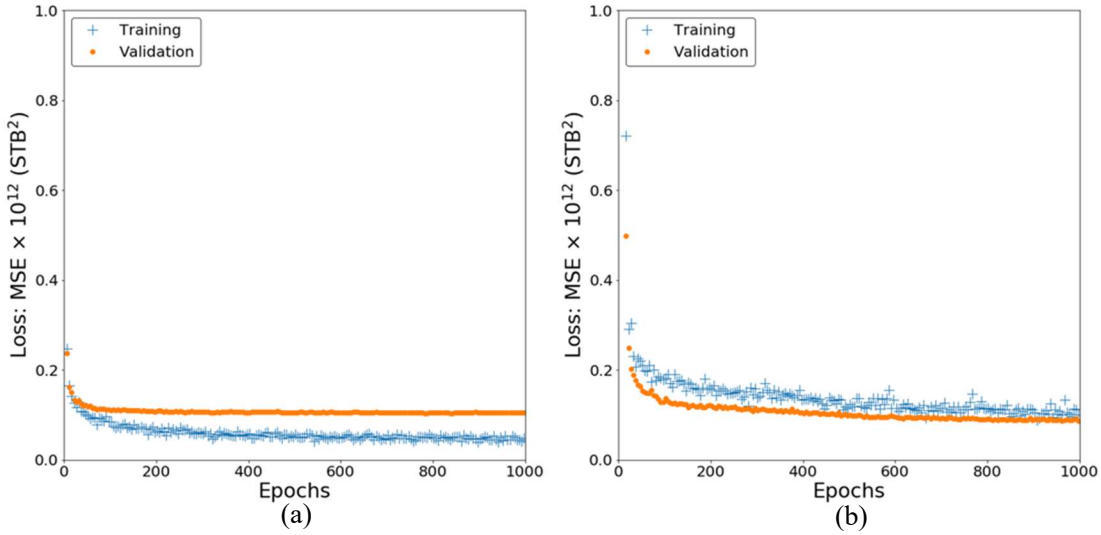


Figure 4.23 Loss function trend at the channelized reservoir (input: permeability + porosity + pressure + oil saturation): (a) ANN and (b) CNN.

4.5.7. 신경망의 예측 결과 요약

Figure 4.24는 채널 저류층의 모든 좌표에 대하여 저류층 시뮬레이션을 수행해 획득한 좌표별 생산성을 나타낸다. 완전히 일치하지는 않지만 생산성이 높은 지점이 채널의 형상과 유사하게 나타나는 것을 확인할 수 있다. 단, 근해 저류층 사례와 마찬가지로 이러한 정답을 실제 저류층 모델을 사용하여 획득하는 것은 불가능에 가깝다.

Figure 4.25와 Figure 4.26은 각각 학습된 인공신경망과 합성곱 신경망을 사용해 채널 저류층 전체에 대해 추가정의 좌표에 따른 생산성을 예측한 그림이다. 근해 저류층의 결과와 유사하게 인공신경망으로 예측한 최대 생산량값은 학습 자료의 최대 범위보다 훨씬 높게 나타나는 사례가 발생하였다(Figure 4.25(a), Figure 4.25(c)). 단, 이러한 경향은 네 가지 입력값을 모두 학습한 인공신경망으로 예측한 경우에는 발생하지 않았다(Figure 4.25(d)). 오일 포화도만을 학습한 경우에는 인공신경망과 합성곱 신경망 모두 명확한 최적의 위치를 도출하지 못 했다(Figure 4.25(b)와 Figure 4.26(b)). 합성곱 신경망의 예측 결과에서는 입력값의 종류가 증가할수록 생산성이 낮은 부분과 높은 부분의 경계가 명확하게 드러나고 있다. 이러한 경향은 인공신경망에서도 나타나지만 Figure 4.25(d)와 Figure 4.26(d)를 비교할 때, 합성곱 신경망에서 더욱 명확하게 경계를 구분하고 있음을 확인할 수 있다. 단, 채널 사이사이의 구멍과 같은 형태로 나타난 생산성이 낮은 부분에 대해서는 인공신경망과 합성곱

신경망 모두 제대로 예측하지 못 하였다. 이러한 결과는 현실성을 위하여 학습 자료에서 셰일 지역에 추가정이 위치하는 경우를 모두 배제했기 때문에 발생한 것으로 해석된다.

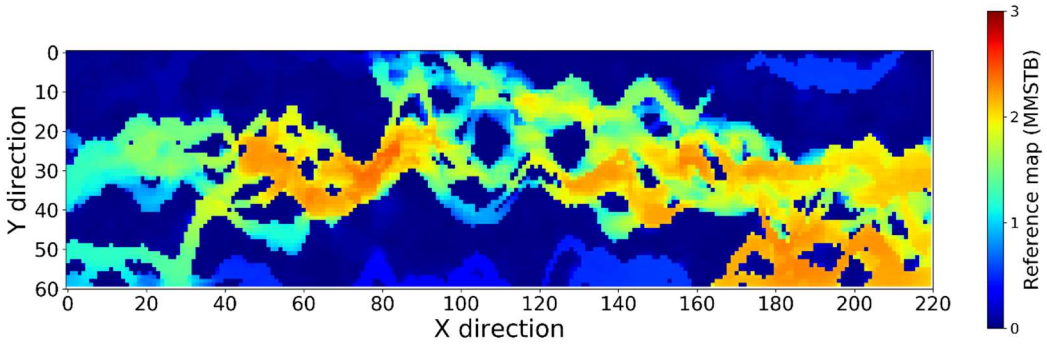


Figure 4.24 Reference productivity map of the SPE10 channelized reservoir.

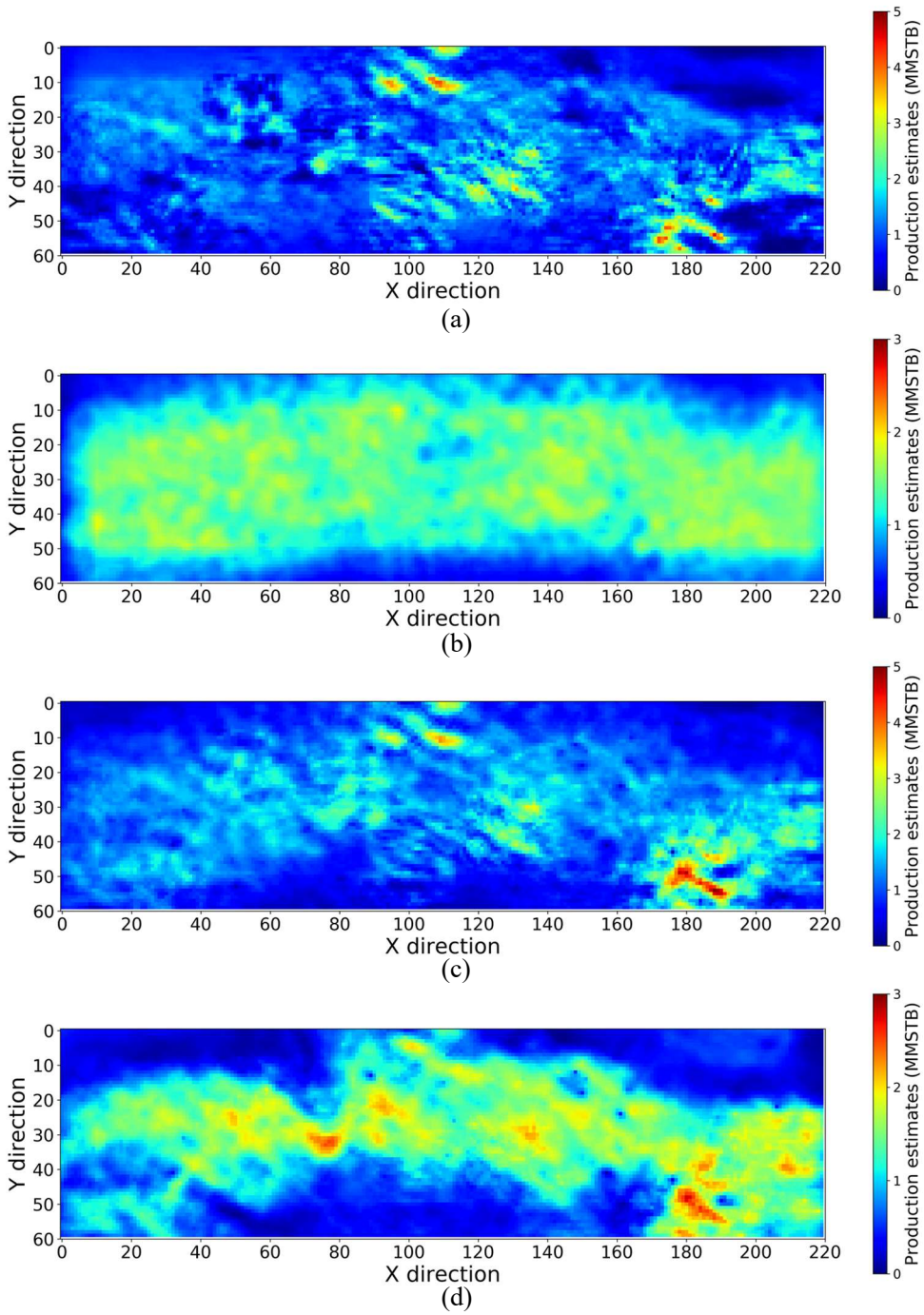


Figure 4.25 Productivity maps of the SPE10 channelized reservoir using ANN: (a) productivity map obtained using k, (b) productivity map obtained using S_o, (c) productivity map obtained using $k + S_o$, and (d) productivity map obtained using $k + \phi + P + S_o$.

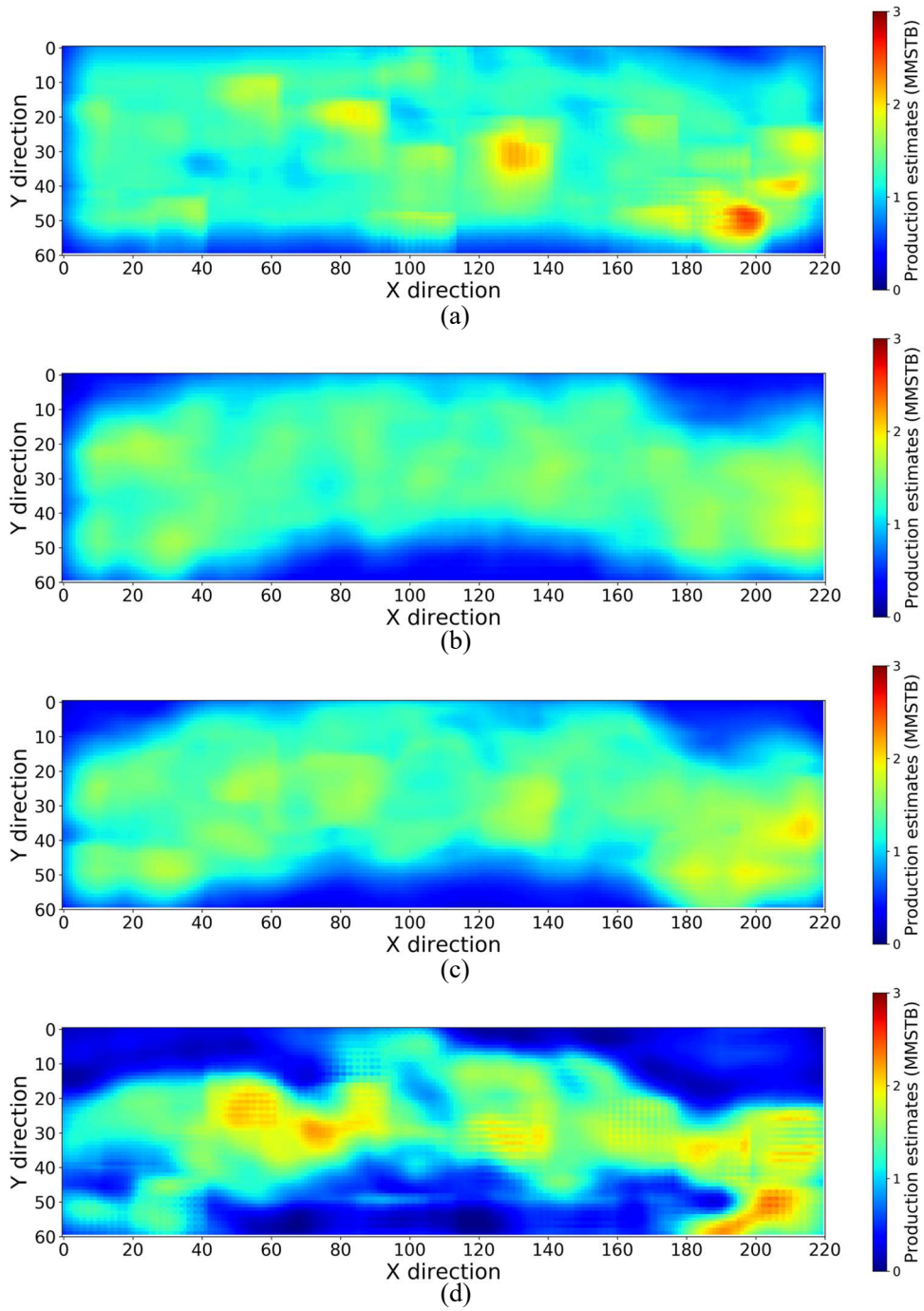


Figure 4.26 Productivity maps of the SPE10 channelized reservoir using CNN: (a) productivity map obtained using k, (b) productivity map obtained using S_o, (c) productivity map obtained using $k + S_o$, and (d) productivity map obtained using $k + \phi + P + S_o$.

Table 4.10은 Figure 4.26(d)의 합성곱 신경망 모델이 예측한 가장 높은 생산성을 가지는 지점 상위 20곳에 대해 실제 저류층 시뮬레이션을 수행한 결과와 상대 오차를 나타낸다. 전체적으로 다소 과대예측을 하는 경향을 보이고 있으며, 14번 사례에서 오차가 1000% 이상 발생하였다. 20개 시나리오의 평균 상대 오차는 9.43%였으나, 14번의 이상 사례(anomaly)를 제외한 경우에는 4.27%이다. 해당 이상 사례들의 좌표를 확인한 결과, 추가정이 채널과 매우 인접하지만 채널 바깥에 위치하는 경우, 즉 추가정을 유체투과도가 매우 낮은 지역에 시추한 사례로 나타났다. 이러한 결과는 사물 인식에서 기계 학습의 한계를 보여주는 것으로, 인간은 직관적으로 해당 지점의 생산성이 낮을 것을 알아차릴 수 있지만 신경망 모델은 그렇지 못하였다.

시뮬레이션을 통한 검증 결과 도출한 최적의 추가정 위치는 약 226만 배럴을 생산하는 3번 사례의 (207, 50)이다. 해당 사례는 합성곱 신경망의 예측값 중에서는 상위 3위였지만, 시뮬레이션으로 확인한 결과 상위 20개 사례 중 가장 높은 생산성을 보이는 것으로 나타났다. 3번에 대한 합성곱 신경망의 예측값은 약 238만 배럴이었으며, 오차는 5.14%이다. 상기한 이상 사례를 제외한다면, 인접한 좌표들의 생산성 예측값은 유사하기 마련이다. 이러한 인접한 좌표들을 하나의 후보로 간주할 때 두 번째 후보 지점은 약 222만 배럴의 생산량을 보이는 16번 사례의 (191, 60)이 되며, 오차는 1.85%이다.

Table 4.10 Comparison of simulation and CNN results for the qualified 20 infill well scenarios at the SPE10 channelized reservoir

Rank	X location	Y location	Predicted production (MMSTB)	Reference production (MMSTB)	Relative error (%)
1	205	52	2.405	2.161	11.29
2	205	50	2.378	2.259	5.25
3	207	50	2.376	2.260	5.14
4	207	52	2.374	2.249	5.58
5	204	52	2.341	2.091	11.99
6	206	52	2.326	2.231	4.25
7	203	52	2.326	2.101	10.73
8	206	50	2.322	2.259	2.81
9	204	50	2.300	2.256	1.95
10	205	51	2.290	2.238	2.32
11	205	54	2.279	2.111	7.97
12	207	51	2.276	2.253	1.03
13	209	50	2.272	2.243	1.31
14	204	51	2.270	0.090	2429.36
15	203	50	2.267	2.240	1.22
16	191	60	2.266	2.225	1.85
17	206	51	2.264	2.253	0.47
18	192	59	2.263	2.253	0.45
19	209	52	2.260	2.245	0.68
20	205	53	2.258	2.122	6.41
Average			2.286	2.130	9.43
Average except for the rank #14			2.308	2.213	4.27

신경망으로 작성한 생산성분포 지도는 시뮬레이션의 사후 정보(posterior information)이다. 이 생산성분포 지도는 유체투과도, 공극률 등 정적 물성의 분포라는 사전 정보(prior information)과 결합함으로써 보다 정교하게 제작할 수 있다. 채널 저류층의 경우, 채널 외부인 셰일층의 낮은 유체투과도로 인하여 채널 내부에서 최적의 추가정 위치가 찾아질 것임이 자명(trivial)하다. Figure 4.27(a)는 유체투과도 인덱스 지도로서 격자별 평균 유체투과도를 기준으로 채널에는 1의 인덱스를, 셰일에는 0의 인덱스를 부여해서 나타낸 그림이다. Figure 4.27(b)는 해당 인덱스 지도와 Figure 4.26(d)의 생산성 지도를 곱해서 획득한 마스킹(masking)된 생산성 지도이다. 따라서 Figure 4.27(b)는 채널 외부의 셰일층의 생산성을 모두 0으로 나타내게 되지만, 앞서의 생산성 지도 Figure 4.26(d)에 비해 참값인 Figure 4.24와 상당히 근접한 형태를 보인다. 이러한 마스킹된 생산성 지도는 의사결정 과정에서 참고 사항으로 유용하게 사용될 수 있다.

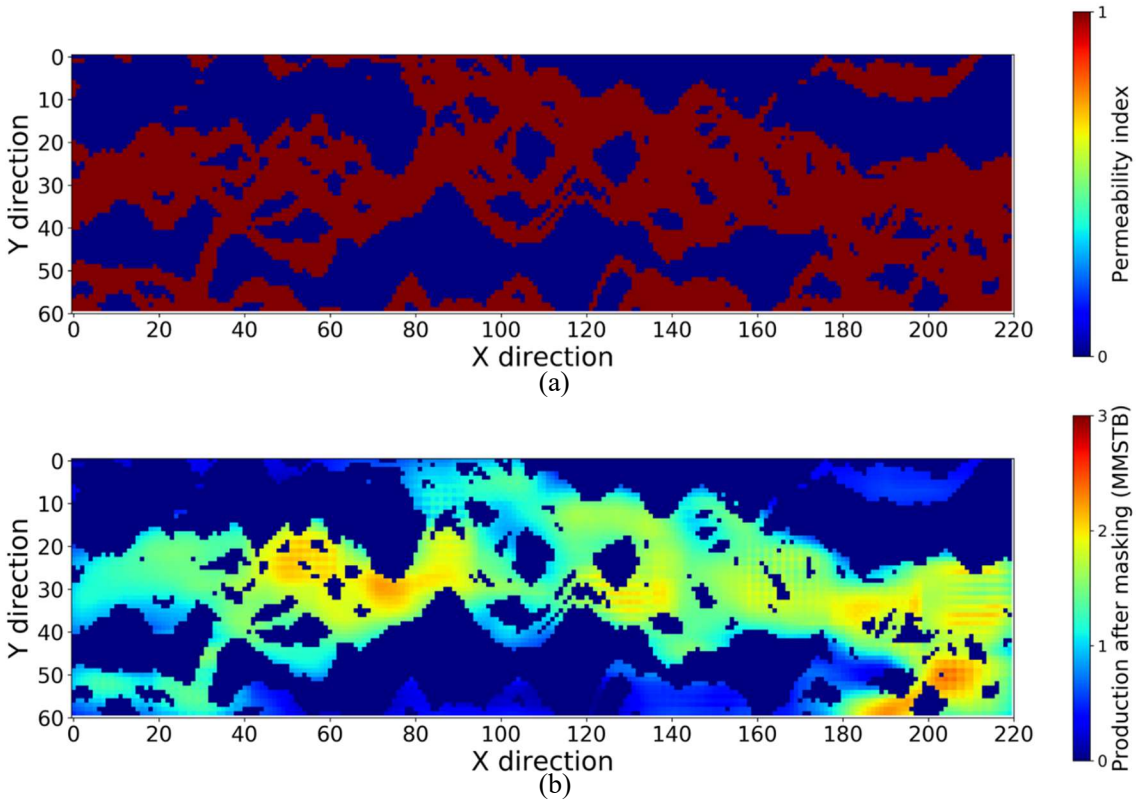


Figure 4.27 (a) permeability index map and (b) masked productivity map obtained using the quad-modal CNN (input: $k + \phi + P + S_o$).

Table 4.11은 Figure 4.25(d)의 인공신경망 모델로 도출한 생산성 상위 20곳의 예측값과 저류층 시뮬레이션의 결과값을 나타낸다. 4.5.3절 및 Table 4.8에서의 결과만으로 인공신경망과 합성곱 신경망의 예측 성능의 차이를 명확히 구분하기는 어렵기 때문에 인공신경망으로 도출한 상위 20개 지점에 대해서도 저류층 시뮬레이션을 통한 검증을 수행하였다. 합성곱 신경망에서의 결과와 유사하게 13번 사례에서 이상 현상이 발생하였다. 합성곱 신경망에 비하여 더 높게 생산성을 예측하는 경향을 보였으며, 20개 사례의 평균 오차는 19.72%였다. 이상 현상을 제외한 19개 사례의 오차는 14.37%였다. Table 4.10의 합성곱 신경망 결과에 비해 오차가 각각 10.29%p, 10.10%p 증가하였다.

Table 4.11 Comparison of simulation and ANN results for the qualified 20 infill well scenarios at the SPE10 channelized reservoir

Rank	X location	Y location	Predicted production (MMSTB)	Reference production (MMSTB)	Relative error (%)
1	49	181	2.769	2.273	21.84
2	49	182	2.737	2.291	19.49
3	50	181	2.718	2.289	18.72
4	52	185	2.632	2.173	21.12
5	50	182	2.578	2.300	12.09
6	51	183	2,555	2.300	11.08
7	48	181	2.553	2.267	12.64
8	49	180	2.549	2.267	12.43
9	49	183	2.544	2.299	10.67
10	53	185	2.543	2.272	11.91
11	33	78	2.525	2.359	7.05
12	48	182	2.518	1.530	64.61
13	53	187	2.511	0.199	1163.30
14	50	180	.2.510	2.130	17.85
15	34	77	2.489	2.320	7.31
16	51	182	2.488	2.299	8.23
17	53	186	2.487	2.176	14.27
18	55	189	2.474	2.293	7.89
19	33	79	2.465	2.353	4.74
20	54	189	2.448	2.290	6.91
Average			2.555	2.134	19.72
Average except for the rank #13			2.557	2.236	14.37

Table 4.12는 네 종류의 입력 자료를 학습한 Figure 4.25(d)의 인공신경망 모델과 Figure 4.26(d)의 합성곱 신경망 모델의 순위 상관(rank correlation)을 나타낸다. 전체 13,200개의 좌표를 유체투과도 15 md를 기준으로 채널과 셰일로 구분한 후, 채널(6,668개 사례)과 셰일(6,532개 사례), 전체(13,200개 사례) 세 개의 영역에 대해서 각각 인공신경망과 합성곱 신경망 모델의 순위 상관을 계산하였다. 또한, 각각의 모델에 대해서 모든 순위에 대해 순위 상관을 구한 경우(No interval), 10의 간격으로 샘플링한 후 순위 상관을 구한 경우(Interval 10), 100의 간격으로 샘플링한 후 순위 상관을 구한 경우(Interval 100)로 구분하여 나타내었다. 순위 상관의 계산 결과, 전체 좌표에 대해서는 합성곱 신경망의 상관도가 미세하게 낮았으며, 채널 부분에 대해서는 합성곱 신경망의 상관도가 인공신경망에 비하여 더 높았다. 실제 학습 자료의 구성에 있어서 관심도가 높은 채널 지역의 사례만을 사용하였음을 감안할 때, 합성곱 신경망의 예측 경향이 인공신경망의 예측 경향에 비하여 더 우수하다고 판단할 수 있다.

Table 4.12 Rank correlation for the quad-modal ANN and CNN

Case (Number of grids)	ANN			CNN		
	No interval	Interval 10	Interval 100	No interval	Interval 10	Interval 100
Channel (6,668)	0.680	0.688	0.652	0.721	0.716	0.789
Shale (6,532)	-0.053	-0.011	-0.151	-0.024	0.010	-0.164
Total (13,200)	0.658	0.647	0.560	0.637	0.626	0.558

Table 4.13은 인공신경망과 합성곱 신경망으로 예측한 상위 사례들이 실제 참값의 상위 사례에 해당하는지 여부를 검토하고 있다. 채널 지역의 총 격자 수인 6,668의 약 5%, 10%, 15%에 해당하는 333, 666, 1,000개의 상위 사례(예측값)가 동일한 크기의 실제 상위 사례(참값)에 포함되는지를 계산하였다. 또한, 학습 자료의 출력값의 범위를 초과하는 생산성을 보이는 영역의 전체인 1,755개의 상위 사례에 대해서도 포함 여부를 계산하였다. 채널 지역의 5%인 상위 333개의 사례에서는 인공신경망과 합성곱 신경망이 예측한 상위 사례의 포함한 개수가 104개로 동일하였다. 즉, 합성곱 신경망으로 예측한 생산성이 높은 상위 333개의 사례 중 실제 생산성이 높은 333개의 사례에 포함되는 사례의 수가 104개라는 의미이다. 채널 지역의 10%와 15%에 해당하는 상위 666개 사례와 상위 1,000개의 사례에서는 합성곱 신경망의 예측 사례가 실제 사례에 포함되는 사례의 수가 각각 244, 385개 였으며, 비율로는 인공신경망의 사례보다 약 10%가량 높았다. 또한, 외삽에 해당하는 학습 자료의 출력값의 범위를 초과하는 영역 전체인 상위 1,755개의 사례에 대해서는 합성곱 신경망이 예측한 상위 사례는 절반 이상이 실제 사례에 포함되었으며, 합성곱 신경망의 예측 사례의 포함 비율이 인공신경망의 사례에 비하여 약 18% 높았다.

Table 4.13 Inclusive proportion for the top reference cases

Case	Top 333	Top 666	Top 1,000	Top 1,755
ANN	104 (31.2%)	219 (32.9%)	358 (35.8%)	766 (43.6%)
CNN	104 (31.2%)	244 (36.6%)	385 (38.5%)	904 (51.5%)

4.6. 저류층 시뮬레이션 대비 소요 시간

본 절에서는 저류층 시뮬레이션 대비 프록시 기법의 소요 시간을 비교하였다. Table 4.14는 이 연구의 저류층 시뮬레이션 및 프록시 연산 수행에 사용한 컴퓨터의 사양을 나타낸다. 이 논문에서 사용한 SPE10 저류층의 CMG 시뮬레이션의 소요 시간은 20년의 생산 기간에 대해서 약 300–500초였다. 소요 시간의 차이는 시뮬레이션의 행렬 연산에 따라 상이할 수 있다. 단, 리스타트(restart) 기능을 사용하여 추가정이 아직 시추되지 않은 초기 10년 동안의 시뮬레이션 과정은 모든 사례에서 공유하도록 설정하였으므로 실질적인 시뮬레이션 소요 시간은 약 100–300초이다. 근해 저류층과 채널 저류층 사이의 유의미한 소요 시간 차이는 발생하지 않았다. 리스타트를 사용한 시뮬레이션의 평균적인 소요 시간을 100초라고 가정하면 전체 13,200개의 좌표에 대해 모두 시뮬레이션을 수행하는데에는 약 15일이 소요되며, 평균 소요 시간을 150초라고 가정하면 약 23일이 소요된다. 따라서 수십-수백만개의 격자로 구성된 현장 규모 저류층의 경우 전체 시나리오에 대한 저류층 시뮬레이션의 수행이 현실적으로 불가할 것이므로, 프록시 개발이라는 이 연구의 당위성을 내포한다.

Figure 4.28은 시뮬레이션의 평균 소요 시간을 150초로 가정할 때의 저류층 시뮬레이션 수행 횟수에 따른 소요 시간 및 신경망 프록시의 사용에 필요한 시간을 나타낸다. 500회와 1,000회, 2,000회의 시뮬레이션 수행에는 각각 0.9일, 1.7일, 3.5일 정도의 시간이 소요된다. 신경망의 구조를 최적화하고 학습과 예측

작업까지 수행하는데 걸린 시간은 경험적으로 하루나 이틀정도였으므로 이를 1.5일로 가정한다면 500개의 학습 자료를 사용한 프록시의 사용에 필요한 시간은 약 2.4일이다. 1,000개와 2,000개의 학습 자료를 사용한다면, 각각 3.2일과 5일 정도가 소요된다. 즉, 15–23일 정도가 소요되는 전체 저류층의 좌표별 생산성 예측 과정을 신경망 프록시 기법을 사용하면 3–5일만에 상당히 정확하게 수행할 수 있다. 따라서 실제 의사결정 과정에서도 제안한 기법은 상당히 유용하게 쓰일 것으로 기대된다.

Table 4.14 Specification of computer resources used in this study

Component	Specification
CPU	i7-7700 @ 3.60 GHz
RAM	16 GB
GPU	NVIDIA GeForce GTX 1070
Operating system	Windows 10 Pro

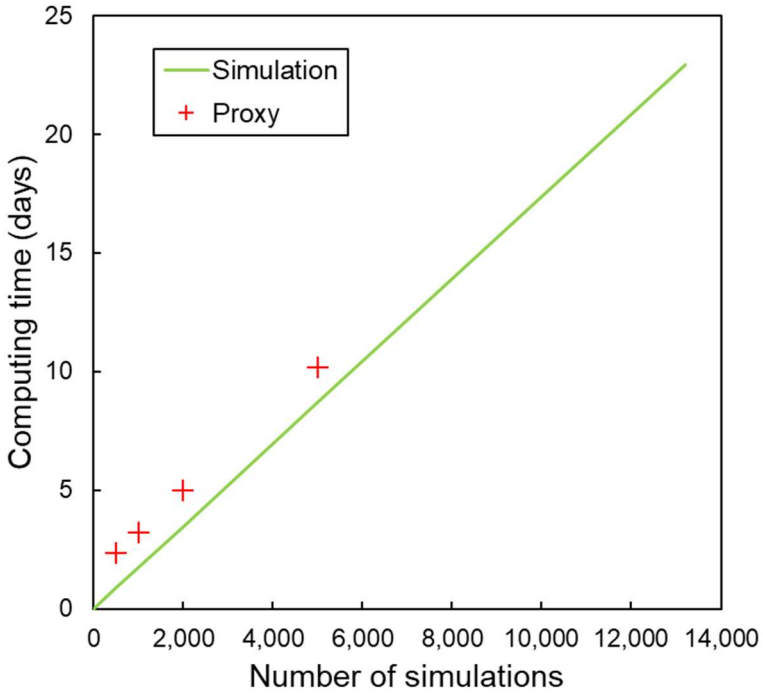


Figure 4.28 Computing time of the simulation and proxy.

4.7. 민감도 분석

본 절에서는 주요 초매개변수의 영향에 대한 민감도 분석을 나타낸다. 합성곱 신경망은 인공신경망에 비해 초매개변수의 종류가 훨씬 다양하다. 또한, 컨볼루션 필터의 초기화 과정에서의 랜덤성이 인공신경망에 비하여 크기 때문에 합성곱 신경망의 수많은 초매개변수에 대한 민감도 분석을 수행하는 것은 매우 어렵다. 이러한 분야에 대해 자동 기계 학습 연구가 활발하게 진행되고 있으나 해당 내용은 본 연구의 영역을 벗어난다. 따라서 이 절에서는 인공신경망과 합성곱 신경망에서 모두 사용되는 초매개변수에 한하여 민감도 분석을 수행하였다. 원활한 분석을 위하여 합성곱 신경망에 비해 상대적으로 일정한 결과를 도출하는 인공신경망을 사용하여 결과를 나타내었다.

4.7.1. 활성화 함수의 영향

Figure 4.29는 신경망 모델의 활성화 함수로 널리 사용되는 sigmoid, tanh, linear, ReLU 함수의 영향을 나타낸다. 4개의 완전연결층으로 구성된 인공신경망의 2개의 은닉층의 활성화 함수로 각각의 함수를 사용하였다. 나머지 출력층에서는 linear 함수를 사용한다. 은닉층을 2개만 사용했음에도 불구하고, sigmoid 함수와 tanh 함수를 사용한 경우는 학습이 전혀 이루어지지 않았고 성능도 매우 낮았다(Figure 4.29(a)와 Figure 4.29(b)). 이는 2.2절에서 언급한 기울기 소실로 인하여 각 층의 출력값이 다음 층으로 원활히 전달되지 않았기

때문이다. Linear 함수를 사용한 경우, 학습 자체가 제대로 진행되지 않았던 sigmoid와 tanh 함수의 경우와는 달리 학습이 상당히 진행되었다(Figure 4.29(c)). 그러나 ReLU 함수를 사용한 Figure 4.29(d)와 비교해서 큰 오차와 낮은 결정계수를 보였다. Linear 함수는 학습 자료의 비선형성을 모사할 수 없기 때문에 은닉층의 수가 늘어나거나 더욱 복잡한 문제에 적용될 경우 ReLU 함수와의 성능 차이가 더욱 증가할 것으로 예상된다.

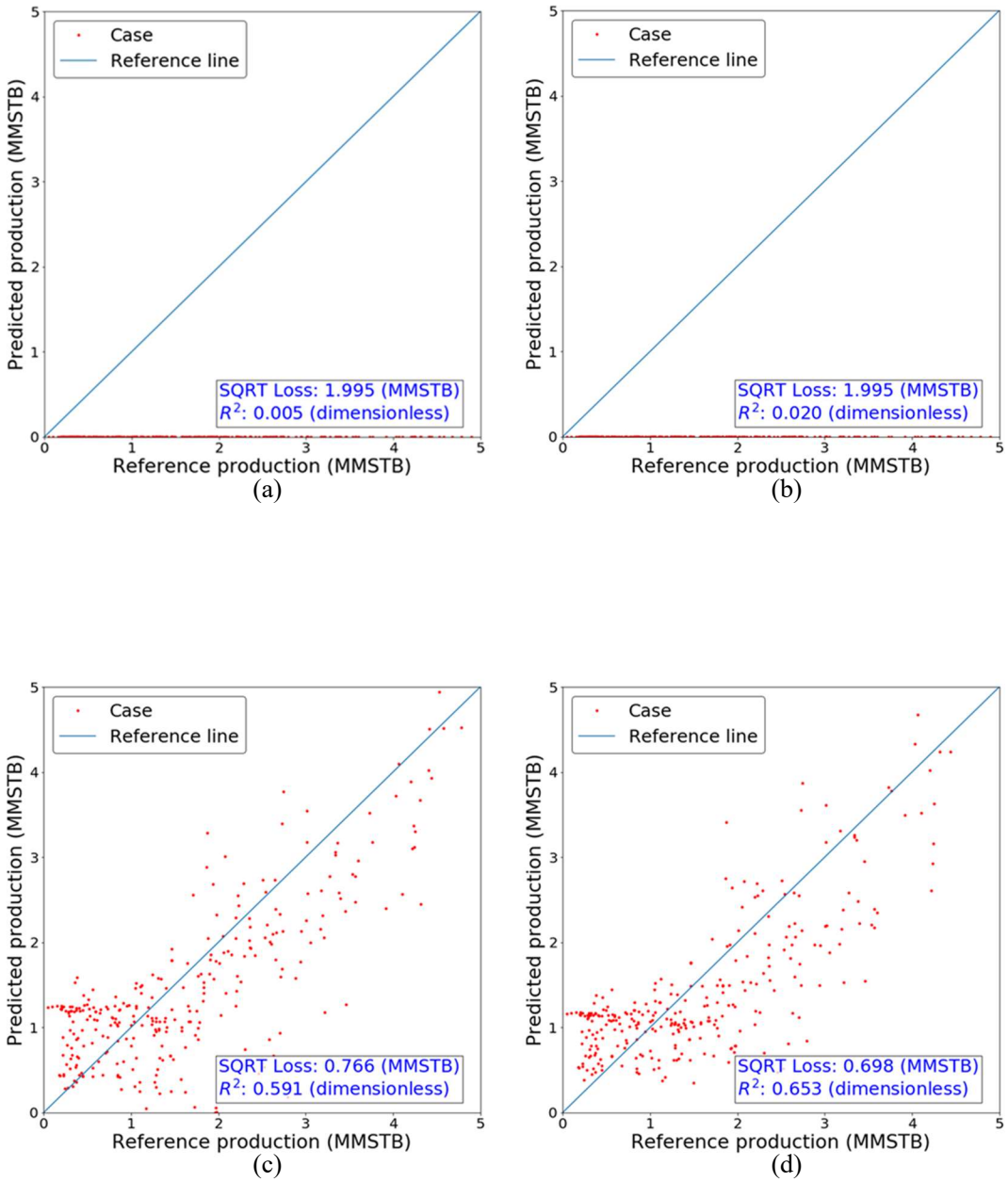


Figure 4.29 Sensitivity analysis of the activation function at the SPE10 shoreface reservoir: (a) sigmoid, (b) tanh, (c) linear, and (d) ReLU.

4.7.2. 학습률의 영향

Figure 4.30은 0.1, 0.01, 0.001, 0.0001의 학습률(learning rate)을 사용하여 유체투과도와 공극률, 압력, 오일 포화도 네 종류의 입력값을 학습한 인공신경망 모델의 시험 성능을 나타낸다. 학습률은 학습 과정에서 손실 함수의 계산을 통해 가중치를 갱신할 때 이전 학습의 영향을 반영하는 비율로서 일종의 보폭을 의미한다. 학습률이 너무 낮다면 가중치의 갱신이 잘 이루어지지 않아 학습 속도가 매우 느리고 국지적 최솟값에 갇힐 위험이 높다. 반대로 학습률이 지나치게 높다면 최적해의 수렴에 어려움을 겪게 된다. 일반적인 경사하강법인 SGD(stochastic gradient descent)에서는 0.1 또는 0.01의 학습률을 주로 사용한다(Zeiler, 2012). 딥러닝에서 사용되는 경사하강법으로는 SGD 외에 이 논문에서 사용하는 Adam과 그 외에 Adagrad, Adamax 등이 있다. 이 논문에서는 Figure 4.30(b)와 같이 0.01의 학습률을 사용한 경우에 모델의 성능이 가장 뛰어났기 때문에 0.01의 학습률을 채택하였다.

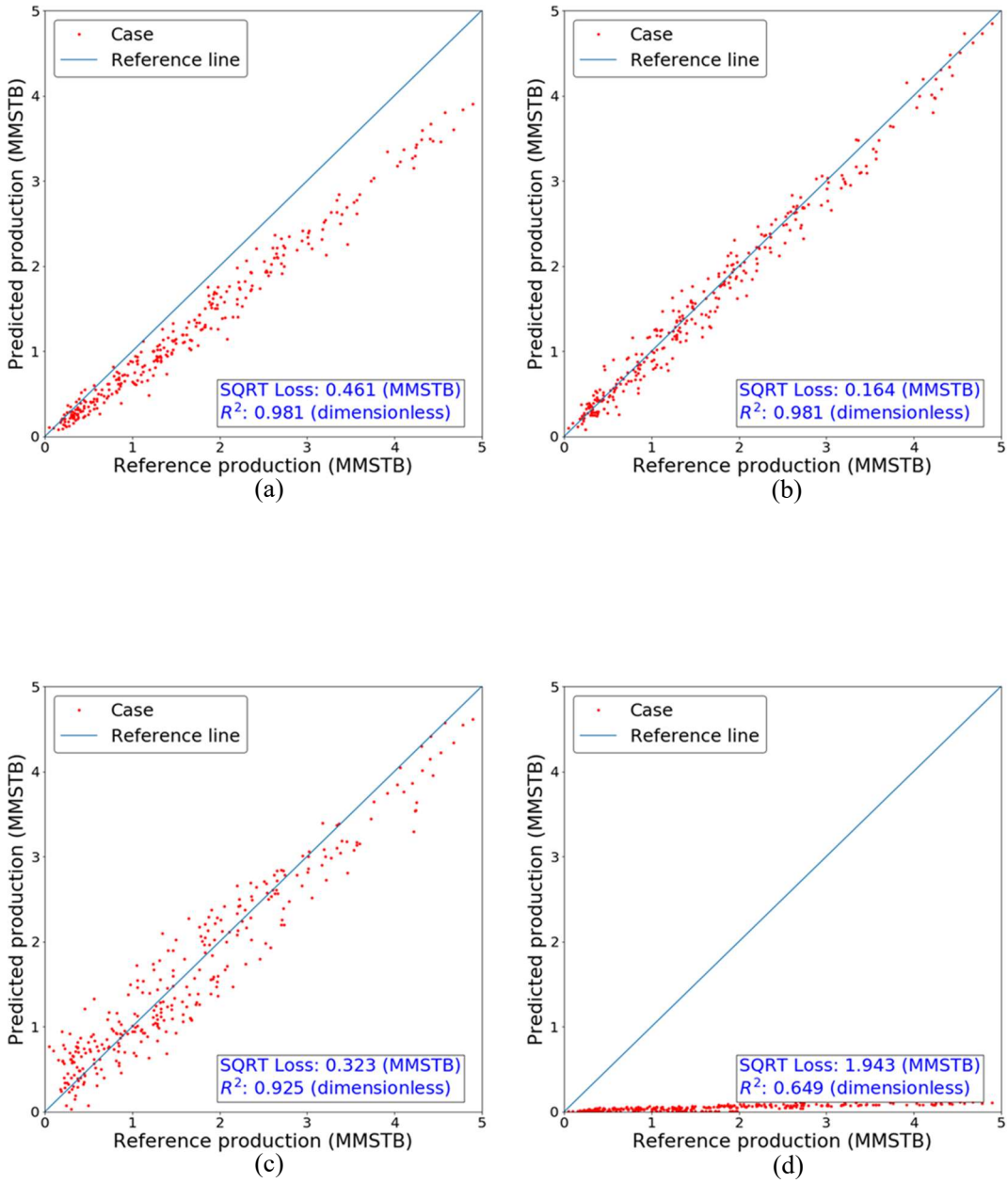


Figure 4.30 Sensitivity analysis of the learning rate at the SPE10 shoreface reservoir: (a) 0.1, (b) 0.01, (c) 0.001, and (d) 0.0001.

4.7.3. 입력 배열의 크기의 영향

Figure 4.31은 각각 1×1×5, 11×11×5, 21×21×5, 31×31×5, 41×41×5, 51×51×5 크기의 배열에 저장된 유체투과도 입력값을 인공신경망으로 학습한 결과이다. 유정의 배유 반경 내 물성 분포를 고려하기 위하여 모든 배열의 중심에 추가정이 위치하며, 인공신경망에 입력되기 때문에 실제로는 1차원 배열로 변환되어 입력된다(예: 11×11×5 → 605×1). 입력 배열의 크기가 커질수록 평균 제곱근 오차는 줄어들고 결정계수는 커지는 경향을 보이므로 모델의 성능은 향상된다고 볼 수 있다. 그러나 동시에 학습에 소요되는 시간과 컴퓨터의 메모리 요구량이 지수 함수적으로 급격히 증가하며, 필요한 학습 자료의 수도 늘어나게 된다.

학습 소요 시간의 경우, Figure 4.31(a)의 1×1×5의 사례에서는 약 50초가 소요되었으며, Figure 4.31(c)의 21×21×5의 사례에서는 약 100초가 소요되었다. 반면에 Figure 4.31(f)의 51×51×5의 사례에서는 300초 이상의 시간이 소요되었으며, 메모리 부족 현상으로 프로그램이 종료되는 경우가 빈번하였다. 입력값의 종류가 늘어나게 되면 학습 시간과 소요 메모리가 더욱 증가하게 되므로 효율적인 연산 수행을 위해 적절한 크기의 입력 배열을 선택해야 한다. 이 논문에서는 유정의 배유 반경을 충분히 포함할 수 있을 것으로 기대되는 동시에 신속하게 학습을 수행하고 입력값의 종류가 늘어나도 무리가 없을 것으로 예상되는 21×21×5 크기의 입력 배열을 선택하였다.

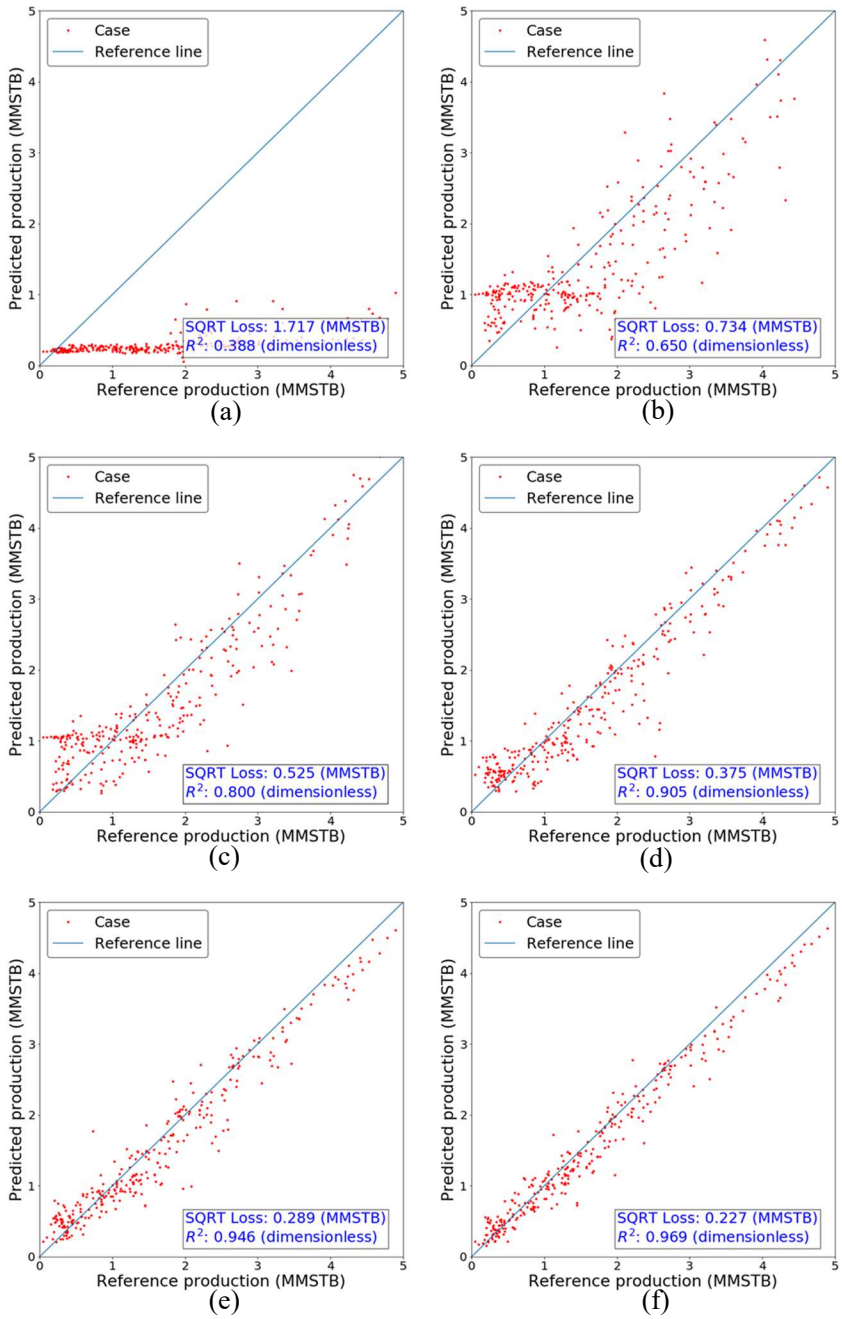


Figure 4.31 Sensitivity analysis of the size of the input array at the SPE10 shoreface reservoir: (a) 1×1×5, (b) 11×11×5, (c) 21×21×5, (d) 31×31×5, (e) 41×41×5, and (f) 51×51×5.

Table 4.14는 560개의 학습 세트를 사용해 학습을 진행하는 근해 저류층 사례에서 인공신경망을 1,000회 학습할 때 소요되는 학습 시간을 입력 자료의 종류 및 크기에 따라 나타낸다. 모든 사례는 GPU를 사용해 연산 속도를 높인 경우에 해당하며, GPU를 사용하지 않을 시에는 적게는 수 배에서 많게는 수십배까지 시간이 소요된다. Table 4.14의 모든 사례는 입력 배열의 크기에 따른 소요 시간을 정확히 나타내고자 모두 동일한 형태의 은닉층 구조를 사용하였다. 그러나 입력 배열의 크기에 비해 은닉층의 노드 수가 지나치게 적은 경우 데이터 병목(bottleneck) 현상이 발생할 위험이 있다. 실제 연구에서는 신경망 구조의 최적화 과정을 통하여 데이터 병목 현상의 위험을 제거한 후 신경망 모델을 사용한다는 것을 감안한다면, 은닉층이 추가될 수 있기 때문에 소요 시간은 Table 4.14의 값보다 늘어날 수 있다. 합성곱 신경망의 소요 시간은 컨볼루션층의 구조에 따라 상당히 유동적이지만, 은닉층보다 컨볼루션층의 연산에 보다 오랜 시간이 소요됨을 감안한다면 일반적으로 인공신경망에 비해 다소 오래 걸린다고 할 수 있다.

Table 4.15 Training time by the size of the input array at the SPE10 shoreface reservoir

ANN case	1×1×5	11×11×5	21×21×5	31×31×5	41×41×5	51×51×5
Single-modal (sec.)	56	67	100	159	235	323
Dual-modal (sec.)	57	79	145	260	418	597
Quad-modal (sec.)	58	104	235	466	768	1119

4.7.4. 은닉층의 개수의 영향

이 절에서는 은닉층의 개수에 따른 모델의 성능 변화를 나타낸다. 은닉층은 인공신경망과 합성곱 신경망에서 동시에 사용되는 구성 요소로서, 다수의 은닉층을 반복적으로 배치함으로써 학습 자료의 입력값과 출력값 사이의 관계를 모사할 수 있다. 단, 은닉층의 수가 늘어날수록 학습가능 매개변수의 수가 대폭적으로 증가하므로 적절한 수의 은닉층을 사용할 필요가 있다. Figure 4.32는 유체투과도, 공극률, 압력, 오일 포화도 네 종류의 입력값을 사용할 때 인공신경망의 은닉층의 개수에 따른 모델의 성능의 변화를 나타낸다. 은닉층의 수가 적정 수준보다 부족한 경우, 시험 세트의 각 사례들은 평균 제곱근 오차는 작으나 결정계수가 높은, 즉 참조선 주위로 수렴하였으나 수렴 수준은 낮은 형태의 결과를 보였다(Figure 4.32(a)). 반면에 은닉층의 수가 과도한 경우, 결정계수가 매우 높아 직선 형태로 수렴하였지만 참조선에서부터 멀리 떨어져 오차가 큰 결과를 보였다(Figure 4.32(c)). 또한, 은닉층의 수가 증가할수록 신경망 모델의 예측값의 출력이 낮아지는 경향을 보였다(Figure 4.32(d)). 이는 과도한 수의 은닉층이 사용되는 경우 ReLU 함수로도 기울기 소실 문제를 완전히 해결할 수 없음을 의미한다. 그러나 적절한 수의 은닉층을 사용한다면 Figure 4.32(b)와 같이 각 사례가 참조선 주위에서 높은 수렴 수준으로 포진하는 결과를 얻을 수 있다.

4.7절에서 기술한 초 매개변수의 민감도 분석과 최적화는 향후 후속연구로서 자동 기계 학습을 결합한 합성곱 신경망을 이용한 유정 위치 최적화에서 다룰 수 있을 것으로 기대한다.

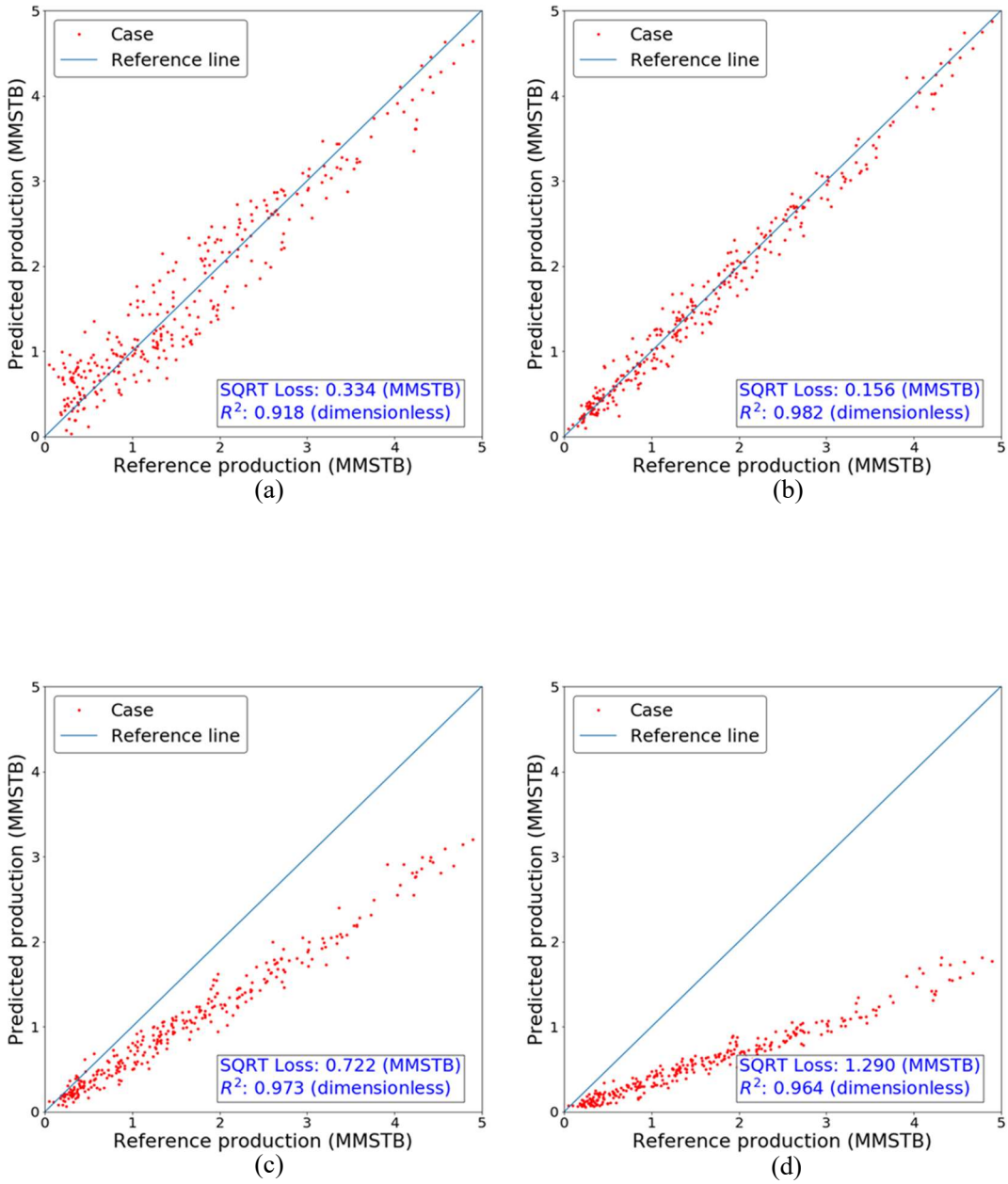


Figure 4.32 Sensitivity analysis of the number of hidden layers at the SPE10 shoreface reservoir: (a) 2 layers, (b) 5 layers, (c) 10 layers, and (d) 15 layers.

5. 결론

이 연구에서 제안한 기법을 SPE10 근해 저류층과 채널 저류층에 적용하여 획득한 연구 결과를 바탕으로 다음과 같은 결론을 도출하였다.

(1) 합성곱 신경망은 기존의 인공신경망과는 달리 다수의 입력값이 나타내는 공간적 분포의 영향을 입력하여 특징추출을 수행할 수 있다. 따라서 유체투과도와 같은 물성의 공간적 분포 특징이 유정의 생산성에 영향을 미치는 경우, 합성곱 신경망은 효과적으로 물성의 분포와 유정의 생산성 사이의 관계를 학습 및 분석할 수 있었다.

(2) 신경망 모델의 성능 향상을 위해 유체투과도나 공극률과 같은 정적 물성 외에 오일 포화도나 압력과 같은 동적 물성의 영향력을 함께 고려할 필요가 있다. 이를 위하여 합성곱 신경망에 다중모달 학습을 적용하여 다수의 입력값을 통합적으로 고려할 수 있도록 하였다. 그 결과 단일 입력값을 사용하는 경우보다 입력값의 종류가 늘어날수록 신경망 모델의 성능이 향상되었다.

(3) 유정 주위의 물성 분포를 인공신경망의 입력 자료로 사용함으로써 유정이 위치한 격자의 물성만을 사용한 기존 연구보다 훨씬 높은 성능을 얻을 수 있었다. 인공신경망은 합성곱 신경망에 비하여 물성의 공간적

분포를 분석하는데 있어서 다소 성능이 떨어지지만, 양질의 학습 자료를 통하여 그러한 한계를 극복할 수 있음을 확인하였다. 또한, 인공신경망은 학습에 소요되는 시간이 상대적으로 짧고 최적의 학습을 위한 초매개변수 설정이 쉽다는 장점이 있다. 따라서 입력 자료의 공간적 분포 패턴이 뚜렷하지 않은 경우나 학습을 신속히 수행해야 하는 경우, 분석 초기의 테스트를 하는 경우 등에 인공신경망은 널리 사용될 수 있다.

(4) 근해 저류층과 채널 저류층 사례 모두에서 합성곱 신경망의 시험 성능이 전체적으로 인공신경망의 시험 성능에 비해 높거나 비슷한 수준을 보였다. 학습된 신경망 모델을 사용해 각각의 저류층에서 모든 좌표에 대한 생산성을 예측하였다. 근해 저류층에서 합성곱 신경망이 도출한 상위 20개 시나리오의 생산성에 대한 오차는 약 1.50%로 매우 우수하였다. 채널 저류층에서 합성곱 신경망이 도출한 상위 20개 시나리오의 생산성에 대한 오차는 약 7.32%였으며, 추가정이 채널 바깥에 위치하는 이상 시나리오를 제외한 오차는 약 4.27%였다. 인공신경망이 도출한 채널 저류층의 상위 20개 시나리오의 생산성에 대한 오차는 이상 시나리오를 제외할 시 약 14.37%였다.

(5) 채널 저류층의 모든 좌표에 대해 채널과 셰일로 구분한 후 합성곱 신경망과 인공신경망을 사용해 순위 상관을 계산하였다. 그 결과 0.680의

상관계수를 보인 인공신경망에 비하여 합성곱 신경망은 0.721의 상관계수를 보였으며, 특히 100의 간격으로 샘플링한 후 계산한 순위 상관의 경우 0.789의 높은 결과를 나타냈다. 이러한 결과를 바탕으로 합성곱 신경망의 예측 경향이 인공신경망의 예측 경향보다 우수하다고 판단하였다.

이 연구에서는 추가정의 최적 위치를 선정하는 과정에서 신속한 의사결정을 위하여 저류층 시뮬레이션 기반의 딥러닝 프록시 기법을 제안하였다. 제안한 다중모달 합성곱 신경망 기법은 복잡한 비선형적 관계를 모사하기 위해 유정 인근의 정적 물성과 동적 물성을 통합하여 유정의 생산성을 학습하였다. 학습된 신경망 모델을 사용하여 학습되지 않은 지점에 대해 유정 인근의 물성 분포만으로 유정의 생산성을 예측할 수 있었다. 이러한 자료기반 기법을 사용함으로써 적은 횟수의 저류층 시뮬레이션 수행만으로 신속하게 최적의 추가정 위치를 선정할 수 있었다. 제안한 기법은 물성의 공간적 분포의 영향력이 중요하거나 다수의 물성 분포를 통합적으로 고려해야 하는 여러 문제에 적용될 수 있다. 향후 유·가스 개발의 다양한 의사 결정 과정에서 제안한 기법이 효율적인 도구로 활용될 것으로 기대한다.

Nomenclature

X	Input array
Y	Output array
$Y_{N^{fc}}$	Final output array
W	Weight array or convolutional filter
B	Bias array
$\hat{Y}$	Reference output array
D	Auxiliary input array
σ	Activation function
f	Process of whole convolutional and pooling layers
g	Process of whole fully connected layers
h	Concatenating function
θ	Set of trainable parameters
J	Loss function
I	Number of input array types
N_{train}	Number of training data
ε	Relative error
cv	Convolution
fc	Full connection
k	Absolute permeability (Permeability)
k_o	Effective oil permeability
k_{ro}	Relative oil permeability
ϕ	Porosity
P	Pressure
S_o	Oil saturation

S_w	Water saturation

참고문헌

S. M. Al-Fattah and R. A. Startzman, “Predicting natural gas production using artificial neural network,” in *SPE Hydrocarbon Economics and Evaluation Symposium*, Dallas, Texas, USA, April 2001. SPE-68593-MS.

G. A. Alusta, E. J. Mackay, J. Fennema, and I. Collins, “EOR vs. Infill well drilling: How to make the Choice?,” in *SPE Enhanced Oil Recovery Conference*, Kuala Lumpur, Malaysia, July 2011. SPE-143300-MS.

J. Asadisaghandi and P. Tahmasebi, “Comparative evaluation of back-propagation neural network learning algorithms and empirical correlations for prediction of oil PVT properties in Iran oilfields,” *Journal of Petroleum Science and Engineering* 78 (2): 464–475, 2011.

S. Behnke, *Hierarchical neural networks for image interpretation*, Springer, vol. 2766, 2003.

A. Centilmen, T. Ertekin, and A. S. Grader, “Applications of neural networks in multiwell field development,” in *SPE Annual Technical Conference and Exhibition*, Houston, Texas, USA, October 1999. SPE-56433-MS.

F. Chollet, *Deep learning with python*, Manning Publications Co., 2017.

M. A. Christie and M. J. Blunt, “Tenth SPE comparative solution project: A comparison of upscaling techniques,” in *SPE Reservoir Simulation Symposium*, Houston, Texas, USA, February 2001. SPE-72469-PA.

CMG, *IMEX user guide*, Computer Modelling Group LTD., 2017.

A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust RGB-D object recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, September 2015.

S. Eskandarian, P. Bahrami, and P. Kazemi, "A comprehensive data mining approach to estimate the rate of penetration: Application of neural network, rule based models and feature ranking," *Journal of Petroleum Science and Engineering* 156: 605–615, 2017.

M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *the 29th Conference on Neural Information Processing Systems*, Montreal, Canada, December 2015.

M. A. Figueiredo, "Adaptive sparseness for supervised learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25 (9): 1150–1159, 2003.

R. B. Gharbi and A. M. Elsharkawy, "Universal neural network based model for estimating the PVT properties of crude oil systems," in *SPE Asia Pacific Oil and Gas Conference*, Kuala Lumpur, Malaysia, April 1997. SPE-38099-MS.

X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *the 14th International Conference on Artificial Intelligence and Statistics*, Fort Lauderdale, Florida, USA, April 2011.

O. Haas and P. S. Hustad, "Machine learning for evaluation of external and internal surface conditions," in *SPE International Oilfield Corrosion Conference and Exhibition*, Aberdeen, Scotland, UK, June 2018. SPE-190895-MS.

D. M. Hawkins, "The problem of overfitting," *Journal of Chemical Information and Computer Sciences* 44 (1): 1–12, 2004.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," https://arxiv.org/abs/1207.0580, 2012.

S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (2): 107–116, 1998.

N. J. Hyne, *Nontechnical guide to petroleum geology, exploration, drilling, and production*, 2nd edition, PennWell Co., 2001.

I. S. Jang, S. Oh, Y. Kim, C. Park, and H. Kang, "Well-placement optimisation using sequential artificial neural networks," *Energy Exploration & Exploitation* 36 (3): 433–449, 2018.

Y. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," https://arxiv.org/abs/1511.06530, 2015.

D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," https://arxiv.org/abs/1412.6980, 2014.

M. M. Korjani, A. S. Popa, E. Grijalva, S. Cassidy, and I. Ershaghi, "Reservoir characterization using fuzzy kriging and deep learning neural networks," in *SPE Annual Technical Conference and Exhibition*, Dubai, UAE, September 2016. SPE-181578-MS.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation* 1 (4): 541–551, 1989.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE* 86 (11): 2278–2324, 1998.

S. M. Luthi and I. D. Bryant, "Well-log correlation using a back-propagation neural network," *Mathematical Geology* 29 (3): 413–425, 1997.

L. Ma, Z. Lu, L. Shang, and H. Li, "Multimodal convolutional neural networks for matching image and sentence," in *International Conference on Computer Vision*, Santiago, Chile, December 2015.

S. Mahmoud, "A novel technique for reducing uncertainties in static reservoir modeling and development infill well placement," in *SPE Kuwait Oil and Gas Show and Conference*, Mishref, Kuwait, October 2015. SPE-175331-MS.

A. L. Mass, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *the 30th International Conference on Machine Learning*, Atlanta, Georgia, USA, June 2013.

M. Maysami, R. Gaskari, and S. D. Mohaghegh, "Data driven analytics in Powder River Basin, WY," in *SPE Annual Technical Conference and Exhibition*, New Orleans, Louisiana, USA, September 2013. SPE-166111-MS.

B. H. Min, C. Park, J. M. Kang, H. J. Park, and I. S. Jang, "Optimal well placement based on artificial neural network incorporating the productivity potential," *Energy Sources, Part A: Recovery, Utilization, and Environmental Effects* 33 (18): 1726–1738, 2011.

J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *the 28th International Conference on Machine Learning*, Bellevue, Washington, USA, June 2011.

K. O`Shea and R. Nash, “An introduction to convolutional neural networks,” https://arxiv.org/abs/1511.08458, 2015.

E. A. Osman, O. A. Abdel-Wahhab, and M. A. Al-Marhoun, “Prediction of oil PVT properties using neural networks,” in *SPE Middle East Oil Show*, Manama, Bahrain, March 2001. SPE-68233-MS.

M. Pennel, J. Hsiung, and V. B. Putcha, “Detecting failures and optimizing performance in artificial lift using machine learning models,” in *SPE Western Regional Meeting*, Garden Grove, California, USA, April 2018. SPE-190090-MS.

G. A. Polizel, G. D. Avansi, and D. J. Schiozer, “Use of proxy models in risk analysis of petroleum fields,” in *SPE EUROPEC featured at the 79th EAGE Conference and Exhibition*, Paris, France, June 2017. SPE-185835-MS.

E. Revelas, B. Sackmann, A. Thurlow, and C. Jones, “Mapping benthic habitat conditions and seafloor deposits using sediment profile imaging and a semiautomated image processing system,” in *Offshore Technology Conference*, Houston, Texas, USA, April 2018. OTC-28878-MS.

F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain,” *Psychological review* 65 (6): 386–408, 1958.

R. Safari, R. Lewis, X. Ma, U. Mutlu, and A. Ghassemi, “Infill-well fracturing optimization in tightly spaced horizontal wells,” *SPE Journal* 22 (2): 582–595, 2017.

A. L. Samuel, “Some studies in machine learning using the game of checkers,” *IBM Journal of Research and Development* 3 (3): 210–229, 1959.

K. Sohn, W. Shang, and H. Lee, “Improved multimodal deep learning with variation of information,” in *the 28th Neural Information Processing Systems*, Montreal, Canada, December 2014.

D. F. Specht, “Probabilistic neural networks,” *Neural Networks* 3 (1): 109–118, 1990.

N. Srivastava and R. R. Salakhutdinov, “Multimodal learning with deep Boltzmann machines,” in *the 26th Neural Information Processing Systems*, Lake Tahoe, Nevada, USA, December 2012.

G. O. Stone, “An analysis of the Delta rule and the learning of statistical associations,” *Parallel Distributed Processing: Explorations in the Microstructure of Cognition* 1: 444–459, 1986.

S. Sundararaman, R. Thethi, and M. Hill, “Data driven virtual sensors for riser prognostic integrity management,” in *Offshore Technology Conference*, Houston, Texas, USA, April 2018. OTC-28732-MS.

S. Wang and S. Chen, “Evaluation and prediction of hydraulic fractured well performance in Montney Formations using a data-driven approach,” in *SPE Western Regional Meeting*, Anchorage, Alaska, USA, May 2016. SPE-180416-MS.

Y. Xue, L. Cheng, J. Mou, and W. Zhao, “A new fracture prediction method by combining genetic algorithm with neural network in low-permeability reservoirs,” *Journal of Petroleum Science and Engineering* 121: 159–166, 2014.

B. Yeten, L. J. Durlofsky, and K. Aziz, “Optimization of nonconventional well type, location, and trajectory,” *SPE Journal* 8 (3): 200–210, 2003.

M. D. Zeiler, “ADADELTA: An adaptive learning rate method,” https://arxiv.org/abs/1212.5701, 2012.

G. Zhu, L. Zhang, P. Shen, and J. Song, “Multimodal gesture recognition using 3-D convolution and convolutional LSTM,” *IEEE Access* 5: 4517–4524, 2017.

Appendix A. Example of a CMG IMEX simulation file for SPE10

```
RESULTS SIMULATOR IMEX 201710
**************************************************************************
** I/O Control Section                                                  **
**************************************************************************
TITLE1 'SPE10 Channel with aquifer by MG Chu'


*INUNIT *FIELD
INTERRUPT RESTART-STOP
WPRN WELL 10
WPRN GRID TIME
WPRN ITER BRIEF
OUTPRN WELL BRIEF
OUTPRN RES NONE
OUTPRN TABLES NONE
OUTPRN GRID BPP IMEXMAP PRES SO SW
WSRF GRID TIME
WSRF WELL 1
RESTART_SR2 SEPARATE
OUTSRF GRID BPP FLUX POROS PRES SO SW


**************************************************************************
    ** Reservoir Description Section                                    **
**************************************************************************
*GRID VARI 60     220     5
```

```
*KDIR DOWN
*DI IVAR
60*50
*DJ JVAR
220*50
*DK ALL
66000*10
*DTOP
13200*12000

**    0 = null block, 1 = active block
NULL CON      1
*CPOR      3.0E-6        ** Rock compressibility and AQUIFER BOTTOM
*AQPROP
**thickness    porosity    permeability    radius      angle
30          0.3              50          100             0
*AQLEAK    OFF
*AQMETHOD    CARTER-TRACY
*PRPOR 14.7           ** reference pressure.

*INCLUDE 'spe_phi_cmg_flu_5layers.dat'
*INCLUDE 'spe_perm_x_cmg_flu_5layers.dat'
PERMJ    EQUALSI
PERMK    EQUALSI * 0.1
**    0 = pinched block, 1 = active block
PINCHOUTARRAY CON                    1
** Solve three equations.
```

```
MODEL OILWATER

**************************************************************************
** Component Property Section                                           **
**************************************************************************
PVT EG 1
**          p          Rs          Bo          Eg        viso        visg
       14.7000        0.0           1     6.00000     1.04000     0.00800
        4800.0      1.0e-2   1.0009895    10.00000     1.04000     0.00810
BWI 1.0
CO 5.0e-6
CVO 0.000E-5
CVW 0.0
CW 3.15E-6
DENSITY OIL 46.244
DENSITY WATER 62.419
REFPW 14.7
VWI 0.31
DENSITY GAS 0.0647

*ROCKFLUID
**************************************************************************
** Rock-Fluid Property Section                                          **
**************************************************************************
*RPT 1
*SWT *SMOOTHEND *OFF
** SW      Krw       Krow
```

```
0.2030    0.0000    0.8000
0.2316    0.0001    0.5852
0.3801    0.0008    0.4040
0.4687    0.0062    0.2562
0.5572    0.0260    0.1419
0.6458    0.0793    0.0612
0.7343    0.1973    0.0139
0.7970    0.4000    0.0000
0.8229    0.4265    0.0000
0.9114    0.8314    0.0000
1.0000    1.0000    0.0000

*INITIAL
*****************************************************************************
** Initial Conditions Section                                              **
*****************************************************************************
USER_INPUT

GOC_PC 0
WOC_PC 0
*PRES *ALL
66000*6000
*SW *ALL
66000*0.203

*NUMERICAL
*****************************************************************************
```

```
** Numerical Control Section                                                    **
*******************************************************************************
*DTMAX        100.              ** Maximum time step size
*DTMIN        0.00001
*MAXSTEPS        1000            ** Maximum number of time steps
*NORM *PRESS    250.0            ** Normal maximum changes per time step
*NORM *SATUR        0.20
*AIM    *STAB *AND_THRESH       ** Use stab and thresh hold switching criteria
*ITERMAX 200
*NCUTS    8

*RUN
*******************************************************************************
** Well and Recurrent Data Section                                             **
*******************************************************************************
*DATE 2001 1 1
*DTWELL 1.0

WELL 'P1'
PRODUCER 'P1'
OPERATE    MIN    BHP    4000.0    CONT REPEAT
**OPERATE    MAX    STL    10000.0    CONT REPEAT
GEOMETRY    J    0.0762    0.37    1.0    0.0
** UBA              ff     Status    Connection
** UBA                    wi              Status    Connection
PERF            WI    'P1'
** UBA                    wi              Status    Connection
```

```
30 110 1      10000.0   OPEN      FLOW-TO   'SURFACE'   REFLAYER
30 110 2      10000.0   OPEN      FLOW-TO   1
30 110 3      10000.0   OPEN      FLOW-TO   2
30 110 4      10000.0   OPEN      FLOW-TO   3
30 110 5      10000.0   OPEN      FLOW-TO   4

*DATE 2001 2 1
*DATE 2001 3 1
*DATE 2001 4 1
*DATE 2001 5 1
*DATE 2001 6 1
*DATE 2001 12 1
*DATE 2002 6 1
*DATE 2002 12 1
*DATE 2003 6 1
*DATE 2003 12 1
WRST TIME
*DATE 2004 1 1
*DATE 2008 1 1
*DATE 2012 1 1
*DATE 2016 1 1
*DATE 2020 1 1
*DATE 2021 1 1
STOP
```

Appendix B. Keras source code: mutli-modal CNN for SPE10

부록 B에는 파이썬 기반의 케라스로 작성한 다중모달 합성곱 신경망 코드를 첨부한다. 파이썬에서 주석은 '#'으로 표시하며, 코드의 한 줄이 끝났다는 표시를 따로 하지 않는다(예: 매트랩의 세미콜론). 또한, for문과 같은 loop문의 표시를 들여쓰기와 콜론으로 나타내기 때문에 정확한 들여쓰기가 필요하다. 인덱싱의 순서는 1이 아닌 0부터 계산한다. 코딩은 주피터 노트북(Jupyter notebook)에서 수행하였다.

```
# Load modules
%matplotlib inline
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
import scipy.io as sio
import random
import nibabel as nib
from mlxtend.preprocessing import one_hot
import keras
import h5py

# Load raw data

Nx, Ny, Nz = 60, 220, 5 # number of grid
```

```
# Load permeability
f = open('spe_perm_x_cmg_shore_5layers.dat')
prop_flu1 = [float(num) for num in f.read().split()]
prop_flu1_norm = np.reshape(prop_flu1, [Nx, Ny, Nz], 'F')
prop_flu1 = np.reshape(prop_flu1, [Nx, Ny, Nz], 'F')
print(prop_flu1_norm.shape)

# porosity
f = open('spe_poro_cmg_modified2_shore_5layers.dat')
prop_flu2 = [float(num) for num in f.read().split()]
prop_flu2_norm = np.reshape(prop_flu2, [Nx, Ny, Nz], 'F')
prop_flu2 = np.reshape(prop_flu2, [Nx, Ny, Nz], 'F')
print(prop_flu2_norm.shape)

# pressure
f = open('spe_pressure_10year_cmg_shore_aquifer_5layers.dat')
prop_flu3 = [float(num) for num in f.read().split()]
prop_flu3_norm = np.reshape(prop_flu3, [Nx, Ny, Nz], 'F')
prop_flu3 = np.reshape(prop_flu3, [Nx, Ny, Nz], 'F')
print(prop_flu3_norm.shape)

# oil saturation
f = open('spe_SO_10year_cmg_shore_aquifer_5layers.dat')
prop_flu4 = [float(num) for num in f.read().split()]
prop_flu4_norm = np.reshape(prop_flu4, [Nx, Ny, Nz], 'F')
prop_flu4 = np.reshape(prop_flu4, [Nx, Ny, Nz], 'F')
```

```
print(prop_flu4_norm.shape)


# mean-std Standardization

prop_flu1_norm = (prop_flu1_norm - np.mean(prop_flu1))/np.std(prop_flu1)

print('prop_flu1 (permeability) :', np.max(prop_flu1), np.min(prop_flu1), np.mean(prop_flu1),
np.std(prop_flu1))

print('prop_flu1_norm:', np.max(prop_flu1_norm), np.min(prop_flu1_norm),
np.mean(prop_flu1_norm), np.std(prop_flu1_norm))

prop_flu2_norm = (prop_flu2_norm - np.mean(prop_flu2))/np.std(prop_flu2)

print('prop_flu2 (porosity) :', np.max(prop_flu2), np.min(prop_flu2), np.mean(prop_flu2),
np.std(prop_flu2))

print('prop_flu2_norm:', np.max(prop_flu2_norm), np.min(prop_flu2_norm),
np.mean(prop_flu2_norm), np.std(prop_flu2_norm))

prop_flu3_norm = (prop_flu3_norm - np.mean(prop_flu3))/np.std(prop_flu3)

print('prop_flu3 (pressure) :', np.max(prop_flu3), np.min(prop_flu3), np.mean(prop_flu3),
np.std(prop_flu3))

print('prop_flu3_norm:', np.max(prop_flu3_norm), np.min(prop_flu3_norm),
np.mean(prop_flu3_norm), np.std(prop_flu3_norm))

prop_flu4_norm = (prop_flu4_norm - np.mean(prop_flu4))/np.std(prop_flu4)

print('prop_flu4 (SO) :', np.max(prop_flu4), np.min(prop_flu4), np.mean(prop_flu4),
np.std(prop_flu4))

print('prop_flu4_norm:', np.max(prop_flu4_norm), np.min(prop_flu4_norm),
np.mean(prop_flu4_norm), np.std(prop_flu4_norm))


# load well location data

f = open('vertical_well_location_cmg.txt')

well_location = [float(num) for num in f.read().split()]

well_location = np.reshape(well_location, [1000, 3])

print(well_location.shape)
```

```
# define trainng data array
dx, dy, dz = 10, 10, 5 # for 21*21*5
num_whole = 1000
num_train = 700
num_test    = 300
nearwell1 = np.zeros([num_train + num_test, 2*dx + 1, 2*dy + 1, dz])
nearwell2 = np.zeros([num_train + num_test, 2*dx + 1, 2*dy + 1, dz])
nearwell3 = np.zeros([num_train + num_test, 2*dx + 1, 2*dy + 1, dz])
nearwell4 = np.zeros([num_train + num_test, 2*dx + 1, 2*dy + 1, dz])
distance    = np.zeros([num_train + num_test, 5])
grid_well_prop = np.zeros([num_train + num_test, 5])
print(nearwell1.shape, nearwell2.shape, nearwell3.shape, nearwell4.shape, distance.shape,
grid_well_prop.shape)

# extract nearwell properties
lx, ly, lz = 50, 50, 10 # size of each grid
existing_x = 29
existing_y = 109

for n in range(well_location[:, 0].size):
    well_x = int(well_location[n, 1]) - 1
    well_y = int(well_location[n, 2]) - 1
    distance[n, 0] = (59 - well_x)*lx
    distance[n, 1] = (well_x - 0)*lx
    distance[n, 2] = (219 - well_y)*ly
    distance[n, 3] = (well_y - 0)*ly
    distance[n, 4] = (((existing_x - well_x)*lx)**2 + ((existing_y - well_y)*ly)**2)**0.5
```

```
    if well_x < dx:
        if well_y < dy:
            for k in range(dz):
                for j in range(0, well_y + dy + 1):
                    for i in range(0, well_x + dx + 1):
                        nearwell1[n, i + (dx - well_x), j + (dy - well_y), k] =
prop_flu1_norm[i, j, k]
                        nearwell2[n, i + (dx - well_x), j + (dy - well_y), k] =
prop_flu2_norm[i, j, k]
                        nearwell3[n, i + (dx - well_x), j + (dy - well_y), k] =
prop_flu3_norm[i, j, k]
                        nearwell4[n, i + (dx - well_x), j + (dy - well_y), k] =
prop_flu4_norm[i, j, k]
        elif Ny - well_y <= dy:
            for k in range(dz):
                for j in range(well_y - dy, Ny):
                    for i in range(0, well_x + dx + 1):
                        nearwell1[n, i + (dx - well_x), j - (well_y - dy), k] =
prop_flu1_norm[i, j, k]
                        nearwell2[n, i + (dx - well_x), j - (well_y - dy), k] =
prop_flu2_norm[i, j, k]
                        nearwell3[n, i + (dx - well_x), j - (well_y - dy), k] =
prop_flu3_norm[i, j, k]
                        nearwell4[n, i + (dx - well_x), j - (well_y - dy), k] =
prop_flu4_norm[i, j, k]
        else:
            for k in range(dz):
                for j in range(well_y - dy, well_y + dy + 1):
                    for i in range(0, well_x + dx + 1):
                        nearwell1[n, i + (dx - well_x), j - (well_y - dy), k] =
prop_flu1_norm[i, j, k]
```

```
                    nearwell2[n, i + (dx - well_x), j - (well_y - dy), k] =
prop_flu2_norm[i, j, k]
                    nearwell3[n, i + (dx - well_x), j - (well_y - dy), k] =
prop_flu3_norm[i, j, k]
                    nearwell4[n, i + (dx - well_x), j - (well_y - dy), k] =
prop_flu4_norm[i, j, k]
    elif Nx - well_x <= dx:
        if well_y < dy:
            for k in range(dz):
                for j in range(0, well_y + dy + 1):
                    for i in range(well_x - dx, Nx):
                    nearwell1[n, i - (well_x - dx), j + (dy - well_y), k] =
prop_flu1_norm[i, j, k]
                    nearwell2[n, i - (well_x - dx), j + (dy - well_y), k] =
prop_flu2_norm[i, j, k]
                    nearwell3[n, i - (well_x - dx), j + (dy - well_y), k] =
prop_flu3_norm[i, j, k]
                    nearwell4[n, i - (well_x - dx), j + (dy - well_y), k] =
prop_flu4_norm[i, j, k]
        elif Ny - well_y <= dy:
            for k in range(dz):
                for j in range(well_y - dy, Ny):
                    for i in range(well_x - dx, Nx):
                    nearwell1[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu1_norm[i, j, k]
                    nearwell2[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu2_norm[i, j, k]
                    nearwell3[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu3_norm[i, j, k]
                    nearwell4[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu4_norm[i, j, k]
        else:
```

```
            for k in range(dz):
                for j in range(well_y - dy, well_y + dy + 1):
                    for i in range(well_x - dx, Nx):
                        nearwell1[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu1_norm[i, j, k]
                        nearwell2[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu2_norm[i, j, k]
                        nearwell3[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu3_norm[i, j, k]
                        nearwell4[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu4_norm[i, j, k]
    else:
        if well_y < dy:
            for k in range(dz):
                for j in range(0, well_y + dy + 1):
                    for i in range(well_x - dx, well_x + dx + 1):
                        nearwell1[n, i - (well_x - dx), j + (dy - well_y), k] =
prop_flu1_norm[i, j, k]
                        nearwell2[n, i - (well_x - dx), j + (dy - well_y), k] =
prop_flu2_norm[i, j, k]
                        nearwell3[n, i - (well_x - dx), j + (dy - well_y), k] =
prop_flu3_norm[i, j, k]
                        nearwell4[n, i - (well_x - dx), j + (dy - well_y), k] =
prop_flu4_norm[i, j, k]
        elif Ny - well_y <= dy:
            for k in range(dz):
                for j in range(well_y - dy, Ny):
                    for i in range(well_x - dx, well_x + dx + 1):
                        nearwell1[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu1_norm[i, j, k]
```

```
                    nearwell2[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu2_norm[i, j, k]
                    nearwell3[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu3_norm[i, j, k]
                    nearwell4[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu4_norm[i, j, k]
        else:
            for k in range(dz):
                for j in range(well_y - dy, well_y + dy + 1):
                    for i in range(well_x - dx, well_x + dx + 1):
                    nearwell1[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu1_norm[i, j, k]
                    nearwell2[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu2_norm[i, j, k]
                    nearwell3[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu3_norm[i, j, k]
                    nearwell4[n, i - (well_x - dx), j - (well_y - dy), k] =
prop_flu4_norm[i, j, k]

# load label (output)
f = open('labels_production_spe10_shore_aquifer_child_20year.txt')
production = [float(num) for num in f.read().split()]
production = np.reshape(production, [-1, 1])
print(production.shape)

# split training & test set
num_whole = 1000
distance_train = np.zeros((num_train, 5))
distance_test = np.zeros((num_test, 5))
production_train = np.zeros((num_train, 1))
```

```
production_test = np.zeros((num_test, 1))


for i in range(num_train + num_test):
    if i < num_train:
        distance_train[i] = distance[i]
        production_train[i] = production[i]
    else:
        distance_test[i - num_train] = distance[i + (num_whole - num_train - num_test)]
        production_test[i - num_train] = production[i + (num_whole - num_train - num_test)]


print(distance_train.shape, distance_test.shape)
print(production_train.shape, production_test.shape)


# Standardization for distance
distance_train = (distance_train - np.mean(distance))/np.std(distance)
distance_test = (distance_test - np.mean(distance))/np.std(distance)
print('Training set:', np.max(distance_train), np.min(distance_train), np.mean(distance_train),
np.std(distance_train))
print('Test set    :    ',          np.max(distance_test),    np.min(distance_test),
np.mean(distance_test),    np.std(distance_test))


# for plot
class AccuracyHistory(keras.callbacks.Callback):
    def on_train_begin(self, logs={}):
        self.acc = []
        self.loss = []
        self.val_acc = []
        self.val_loss = []
```

```
    def on_epoch_end(self, batch, logs={}):
        self.acc.append(logs.get('acc'))
        self.val_acc.append(logs.get('val_acc'))
        self.loss.append(logs.get('loss'))
        self.val_loss.append(logs.get('val_loss'))

history = AccuracyHistory()

# make CNN input
input_col, input_row, input_dep = 2*dx + 1, 2*dy + 1, dz

train_x1 = np.zeros([num_train, input_col, input_row, input_dep])
train_x2 = np.zeros([num_train, input_col, input_row, input_dep])
train_x3 = np.zeros([num_train, input_col, input_row, input_dep])
train_x4 = np.zeros([num_train, input_col, input_row, input_dep])

test_x1 = np.zeros([num_test, input_col, input_row, input_dep])
test_x2 = np.zeros([num_test, input_col, input_row, input_dep])
test_x3 = np.zeros([num_test, input_col, input_row, input_dep])
test_x4 = np.zeros([num_test, input_col, input_row, input_dep])

for i in range(num_train + num_test):
    if i < num_train:
        train_x1[i] = nearwell1[i]
        train_x2[i] = nearwell2[i]
        train_x3[i] = nearwell3[i]
```

```
        train_x4[i] = nearwell4[i]
    else:
        test_x1[i - num_train] = nearwell1[i]
        test_x2[i - num_train] = nearwell2[i]
        test_x3[i - num_train] = nearwell3[i]
        test_x4[i - num_train] = nearwell4[i]

train_x1 = np.reshape(train_x1, [-1, input_col, input_row, input_dep, 1])
train_x2 = np.reshape(train_x2, [-1, input_col, input_row, input_dep, 1])
train_x3 = np.reshape(train_x3, [-1, input_col, input_row, input_dep, 1])
train_x4 = np.reshape(train_x4, [-1, input_col, input_row, input_dep, 1])

test_x1 = np.reshape(test_x1, [-1, input_col, input_row, input_dep, 1])
test_x2 = np.reshape(test_x2, [-1, input_col, input_row, input_dep, 1])
test_x3 = np.reshape(test_x3, [-1, input_col, input_row, input_dep, 1])
test_x4 = np.reshape(test_x4, [-1, input_col, input_row, input_dep, 1])

# 21*21*5 quad-modal CNN
from keras.models import Sequential, Model
from keras.layers import Dense, Conv3D, MaxPooling3D, Flatten, Dropout, concatenate, Merge,
Input, Embedding
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.layers.normalization import BatchNormalization

branch1_input = Input(shape=(input_col, input_row, input_dep, 1), name='main_input1')
branch1 = Conv3D(32, kernel_size=(5, 5, 5), strides=(1, 1, 1), activation='relu', padding='same',
                 input_shape=(branch1_input.shape))(branch1_input)
```

```
branch1 = MaxPooling3D(pool_size=(2, 2, 2), strides=(2, 2, 2))(branch1)
branch1 = Conv3D(64, kernel_size=(5, 5, 1), activation='relu', padding='same')(branch1)
branch1 = MaxPooling3D(pool_size=(2, 2, 2), strides=(2, 2, 2))(branch1)
branch1 = Flatten()(branch1)

branch4_input = Input(shape=(input_col, input_row, input_dep, 1), name='main_input2')
branch4 = Conv3D(32, kernel_size=(5, 5, 5), strides=(1, 1, 1), activation='relu', padding='same',
                 input_shape=(branch4_input.shape))(branch4_input)
branch4 = MaxPooling3D(pool_size=(2, 2, 2), strides=(2, 2, 2))(branch4)
branch4 = Conv3D(64, kernel_size=(5, 5, 1), activation='relu', padding='same')(branch4)
branch4 = MaxPooling3D(pool_size=(2, 2, 2), strides=(2, 2, 2))(branch4)
branch4 = Flatten()(branch4)

branch6_input = Input(shape=(5,), name='distance_input')

branch7_input = Input(shape=(input_col, input_row, input_dep, 1), name='main_input3')
branch7 = Conv3D(32, kernel_size=(5, 5, 5), strides=(1, 1, 1), activation='relu', padding='same',
                 input_shape=(branch4_input.shape))(branch7_input)
branch7 = MaxPooling3D(pool_size=(2, 2, 2), strides=(2, 2, 2))(branch7)
branch7 = Conv3D(64, kernel_size=(5, 5, 1), activation='relu', padding='same')(branch7)
branch7 = MaxPooling3D(pool_size=(2, 2, 2), strides=(2, 2, 2))(branch7)
branch7 = Flatten()(branch7)

branch8_input = Input(shape=(input_col, input_row, input_dep, 1), name='main_input4')
branch8 = Conv3D(32, kernel_size=(5, 5, 5), strides=(1, 1, 1), activation='relu', padding='same',
                 input_shape=(branch4_input.shape))(branch8_input)
branch8 = MaxPooling3D(pool_size=(2, 2, 2), strides=(2, 2, 2))(branch8)
```

```
branch8 = Conv3D(64, kernel_size=(5, 5, 1), activation='relu', padding='same')(branch8)

branch8 = MaxPooling3D(pool_size=(2, 2, 2), strides=(2, 2, 2))(branch8)

branch8 = Flatten()(branch8)


main_branch = concatenate([branch1, branch4, branch7, branch8, branch6_input], axis=1)

main_branch = Dense(3000, activation='relu', use_bias=True,
bias_initializer='zeros')(main_branch)

main_branch = Dropout(0.2)(main_branch)

main_branch = Dense(1000, activation='relu', use_bias=True,
bias_initializer='zeros')(main_branch)

main_branch = Dropout(0.2)(main_branch)

main_branch = Dense(300, activation='relu', use_bias=True,
bias_initializer='zeros')(main_branch)

main_branch = Dropout(0.2)(main_branch)

main_branch = Dense(100, activation='relu', use_bias=True,
bias_initializer='zeros')(main_branch)

main_branch = Dropout(0.2)(main_branch)

main_branch = Dense(30, activation='relu', use_bias=True, bias_initializer='zeros')(main_branch)

main_branch = Dropout(0.2)(main_branch)

main_branch_output = Dense(1, activation='linear')(main_branch)


main_model = Model(inputs=[branch1_input, branch4_input, branch7_input, branch8_input,
branch6_input], outputs=main_branch_output)

print(main_model.summary())


# Model compiling & training

main_model.compile(loss='mse', optimizer=keras.optimizers.Adam(lr=0.01, decay=0.1),
metrics=['accuracy'])

import time

from time import strftime
```

```
start_time = time.time()

main_model.fit([train_x1, train_x2, train_x3, train_x4, distance_train], production_train,
batch_size=100, epochs=1000, verbose=1, validation_split=0.2, callbacks=[history])

end_time = time.time()

processing_time = end_time - start_time

print('start_time: ', start_time)

print('--- %.2f seconds ---' %(processing_time))

now = strftime('%y%m%d-%H%M%S')

print(now)


# Model test

import matplotlib.ticker as ticker

score = main_model.evaluate([test_x1, test_x2, test_x3, test_x4, distance_test], production_test,
verbose=0)

print('Test loss:', score[0])

print('SQRT loss:', np.sqrt(score[0]))


# Save trained model

from keras.models import load_model

from keras.models import model_from_yaml


# Save network weights

main_model.save_weights(‘my_model.h5’)

# save network architecture

model_yaml = main_model.to_yaml()

with open('my_model.yaml', 'w') as yaml_file:

    yaml_file.write(model_yaml)
```

```
# Load saved model

from keras.models import load_model

from keras.models import model_from_yaml


yaml_file = open('my_model.yaml', 'r')

loaded_model_yaml = yaml_file.read()

yaml_file.close()

loaded_model = model_from_yaml(loaded_model_yaml)


# Load weights into new model

loaded_model.load_weights('my_model.h5')

print('Loaded model from disk')

loaded_model.compile(loss='mse', optimizer=keras.optimizers.Adam(lr=0.01, decay=0.1),
metrics=['accuracy'])
```

Appendix C. Visualization of feature maps

쿼드모달 합성곱 신경망의 컨볼루션 분기의 각 층(컨볼루션층, 풀링층)에서 도출되는 특징 지도를 시각화하여 나타낸다. 영상 인식 분야에서 널리 사용되는 MNIST나 CIFAR-10(Canadian Institute For Advanced Research) 데이터셋은 모두 2차원 이미지이기 때문에 컨볼루션과 풀링도 2차원으로 진행된다. 그러나 이 연구에서는 입력값이 모두 3차원 배열이므로 3차원의 컨볼루션과 풀링을 적용하였다. 이러한 3차원 배열의 시각화를 위해서는 2차원 평면을 하나씩 나타낼 수 밖에 없다. 따라서 컨볼루션과 풀링의 출력값이 입력값과 상당히 달라보이는 경우가 발생하지만 이는 3차원의 컨볼루션과 풀링 과정때문임을 미리 밝힌다.

Figure D.1과 Figure D.2는 각각 근해 저류층과 채널 저류층 사례에서 사용된 학습 자료 하나씩을 나타낸다. 각각의 학습 자료는 유체투과도, 공극률, 압력, 오일 포화도의 네 종류의 물성으로 구성되며, 각각의 물성은 21×21×5의 크기로 구성된다. 모든 물성은 표준화 과정을 거쳐 0의 평균과 1의 표준편차를 가지는 분포로 전환되었으며, 따라서 단위는 존재하지 않는다. 유체투과도의 경우 로그분포를 취하지 않았기 때문에 편차가 매우 크게 나타난다.

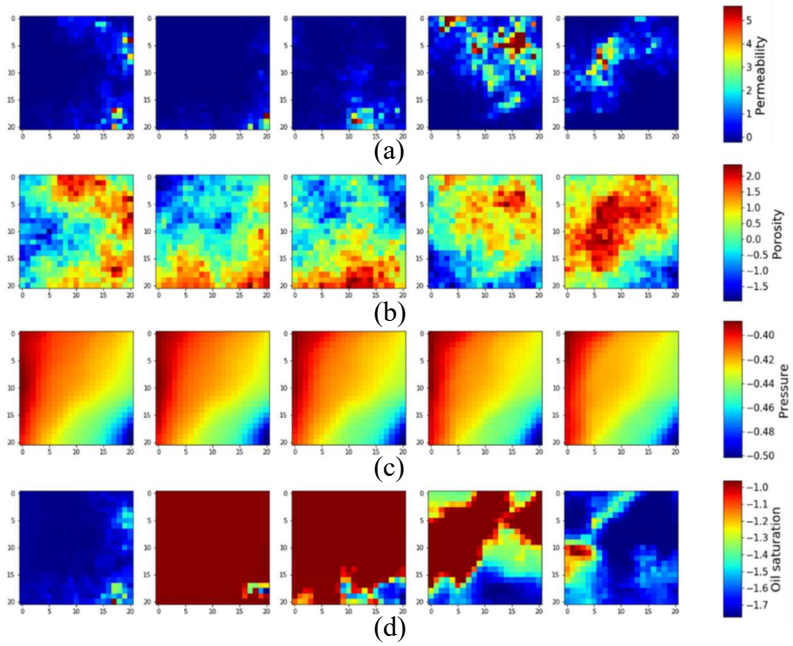


Figure D.1 Input arrays for CNN at the shoreface reservoir: (a) permeability, (b) porosity, (c) pressure, and (d) oil saturation.

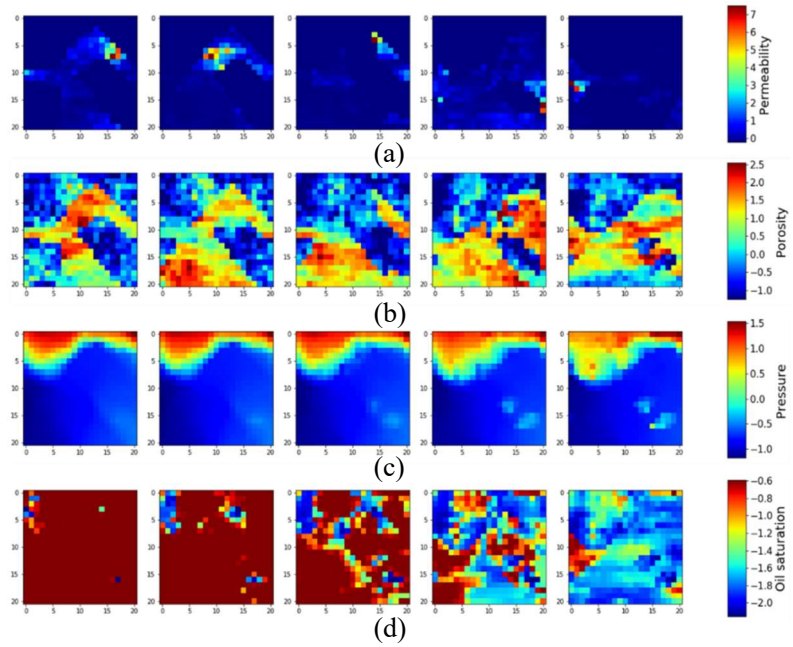


Figure D.2 Input arrays for CNN at the channelized reservoir: (a) permeability, (b) porosity, (c) pressure, and (d) oil saturation.

Figure D.3과 Figure D.4는 근해 저류층에서 유체투과도를 입력값으로 사용한 컨볼루션 분기의 첫 번째 컨볼루션층과 두 번째 컨볼루션층의 특징 지도를 나타낸다. 근해 저류층 사례에서는 제로패딩을 사용하지 않았기 때문에 컨볼루션 과정을 거칠 때마다 출력되는 특징 지도의 크기가 줄어들게 된다. 이 연구에서는 첫 번째 컨볼루션 과정에서 출력값의 3차원 인덱스가 1이 되도록 설정하였으므로 이후 모든 층에서의 출력값의 2차원으로 구성된다. 첫 번째와 두 번째 컨볼루션층에서 각각 20개의 컨볼루션 필터를 적용했기 때문에 각 층에서는 20개씩의 특징 지도가 도출된다.

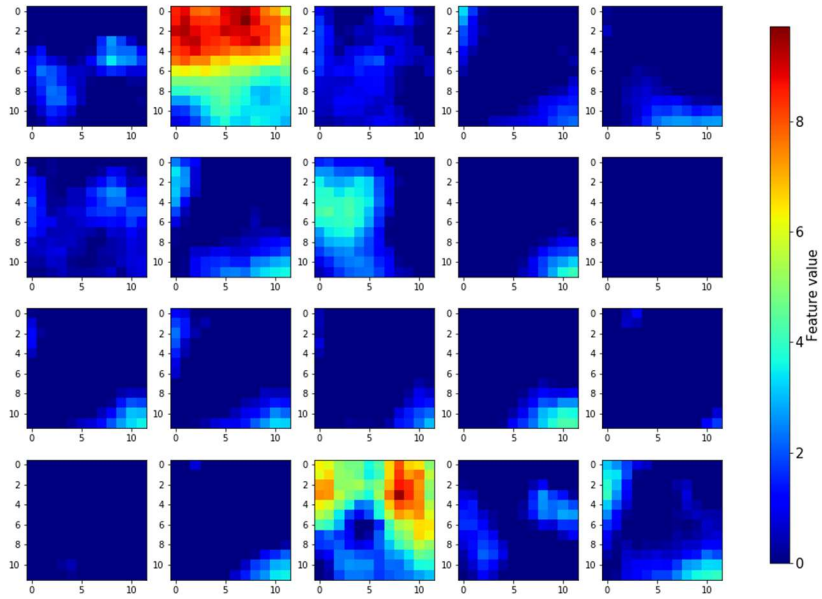


Figure D.3 Feature maps of the first convolutional layer at the shoreface reservoir (input: permeability).

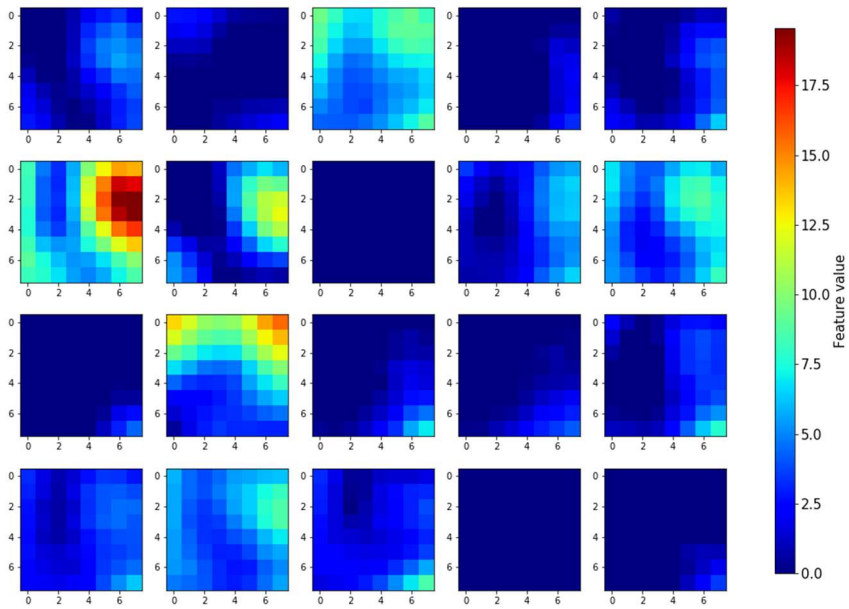


Figure D.4 Feature maps of the second convolutional layer at the shoreface reservoir (input: permeability).

Figure D.5부터 Figure D.7까지는 각각 근해 저류층에서 공극률과 압력, 오일 포화도를 입력값으로 사용한 컨볼루션 분기의 첫 번째 컨볼루션층의 특징 지도를 나타낸다. 입력값의 분포 자체가 거의 동일한 값을 가지는 압력에 대한 특징 지도는 각각의 값은 다르지만 패턴의 형태가 아닌 하나의 값을 전체 배열이 가지는 형태로 도출되었다.

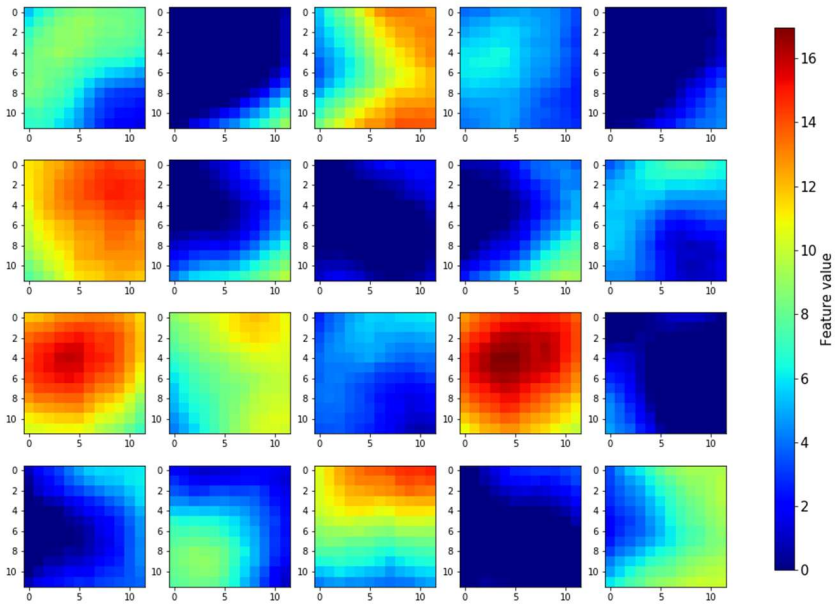


Figure D.5 Feature maps of the first convolutional layer at the shoreface reservoir (input: porosity).

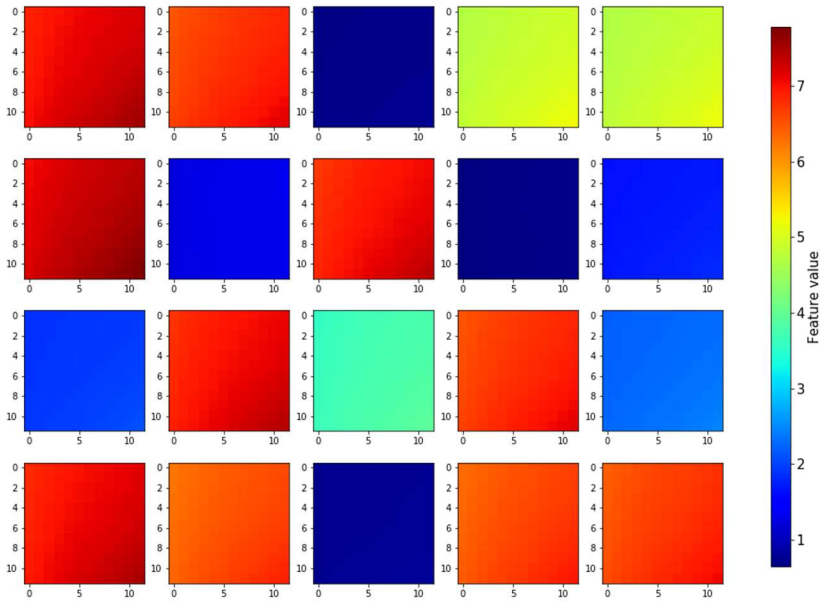


Figure D.6 Feature maps of the first convolutional layer at the shoreface reservoir (input: pressure).

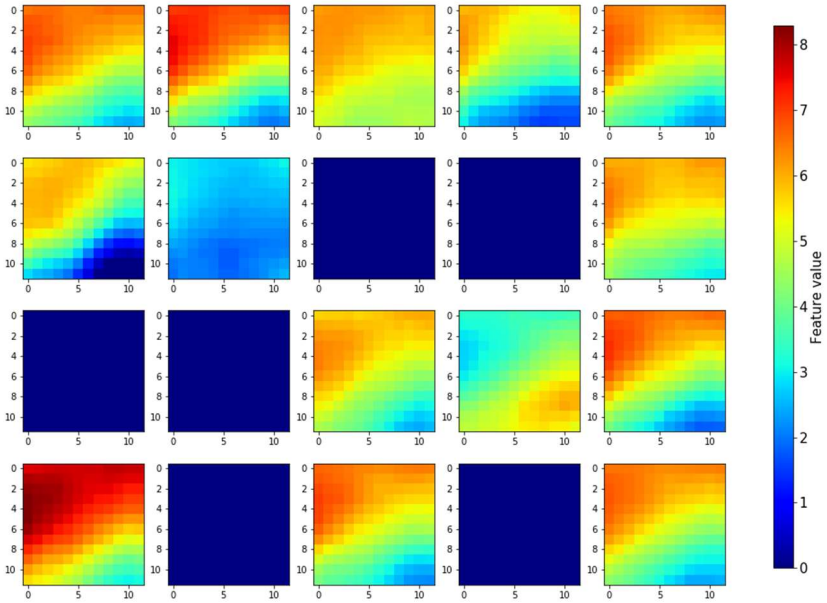


Figure D.7 Feature maps of the first convolutional layer at the shoreface reservoir (input: oil saturation).

Figure D.8은 채널 저류층에서 유체투과도를 입력값으로 사용한 컨볼루션 분기의 첫 번째 컨볼루션층의 특징 지도의 다섯 층 중에서 첫 번째 층을 나타낸다. 채널 저류층 사례에서는 제로패딩을 적용했기 때문에 컨볼루션 과정에서의 출력값의 크기에 변화가 없다. 따라서 도출되는 특징 지도는 21×21×5의 크기를 가지며, 컨볼루션 필터의 수만큼 100개로 구성된다.

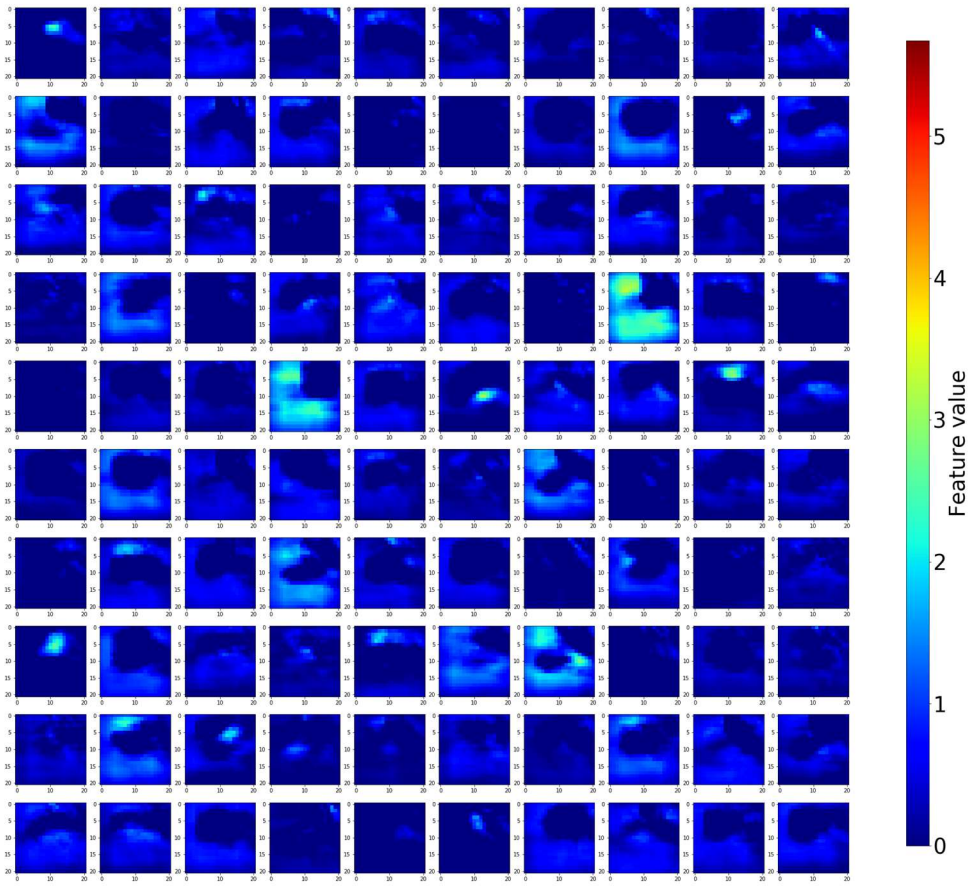


Figure D.8 The first layer of feature maps of the first convolutional layer at the channelized reservoir (input: permeability).

Figure D.9는 채널 저류층에서 유체투과도를 입력값으로 사용한 컨볼루션 분기의 첫 번째 풀링층의 특징 지도의 두 개의 층 중에서 첫 번째 층을 나타낸다. (2, 2, 2)의 맥스풀링을 적용한 첫 번째 풀링층의 특징 지도의 크기는 10×10×2가 되며, 특징 지도의 수에는 변함이 없다. 풀링층의 입력값의 크기와 풀링의 크기의 비율이 정수가 되지 않는 경우, 나머지 부분은 마지막 풀링 과정에 산입된다.

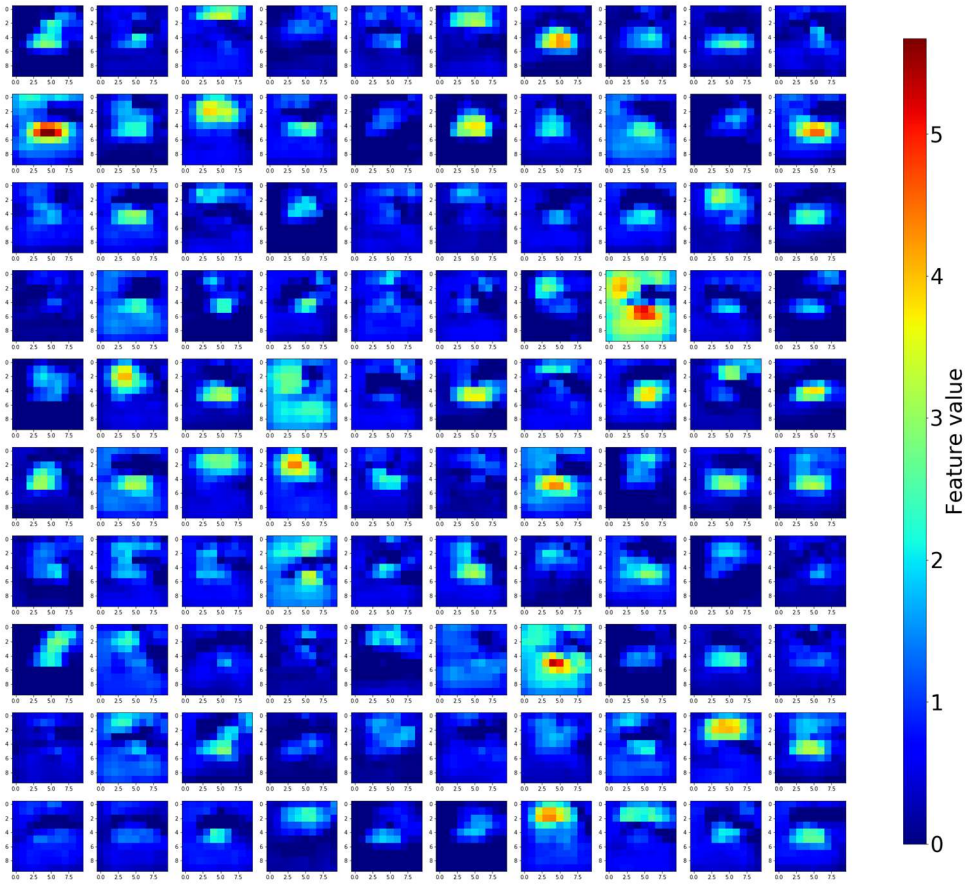


Figure D.9 The first layer of feature maps of the first maxpooling layer at the channelized reservoir (input: permeability).

Figure D.10부터 Figure D.12까지는 각각 채널 저류층에서 공극률과 압력, 오일 포화도를 입력값으로 사용한 컨볼루션 분기의 첫 번째 컨볼루션층의 특징 지도의 첫 번째 층을 나타낸다.

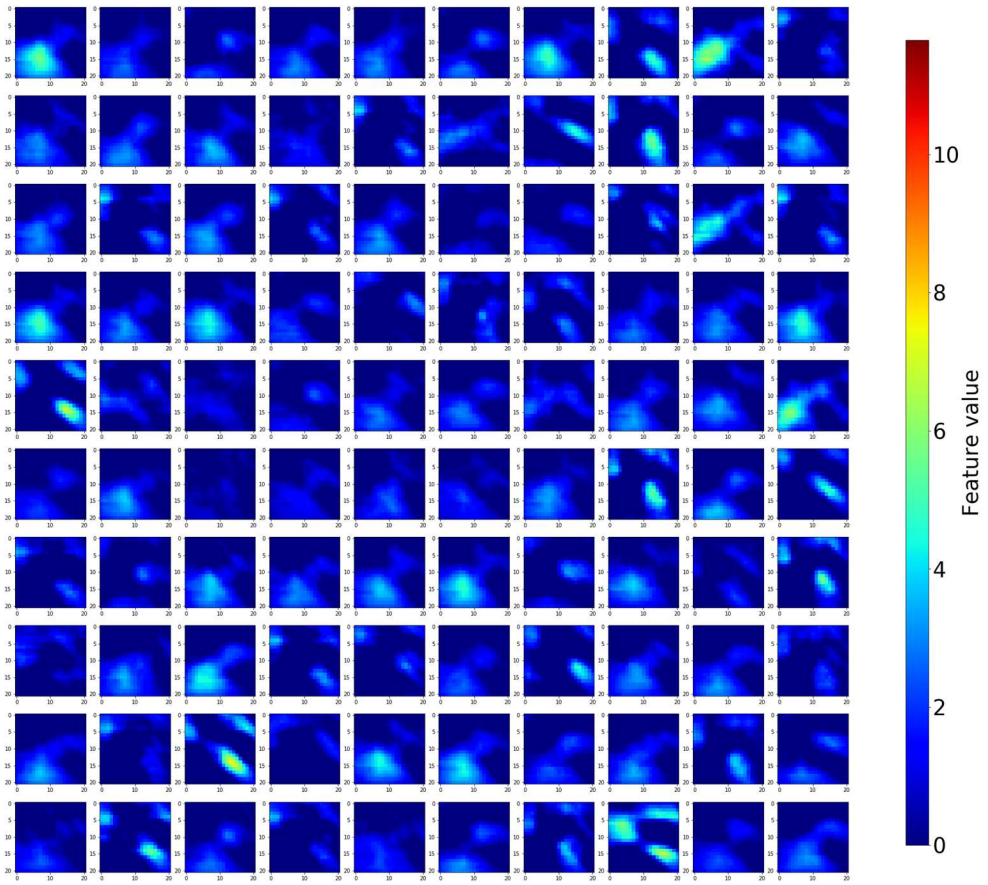


Figure D.10 The first layer of feature maps of the first convolutional layer at the channelized reservoir (input: porosity).

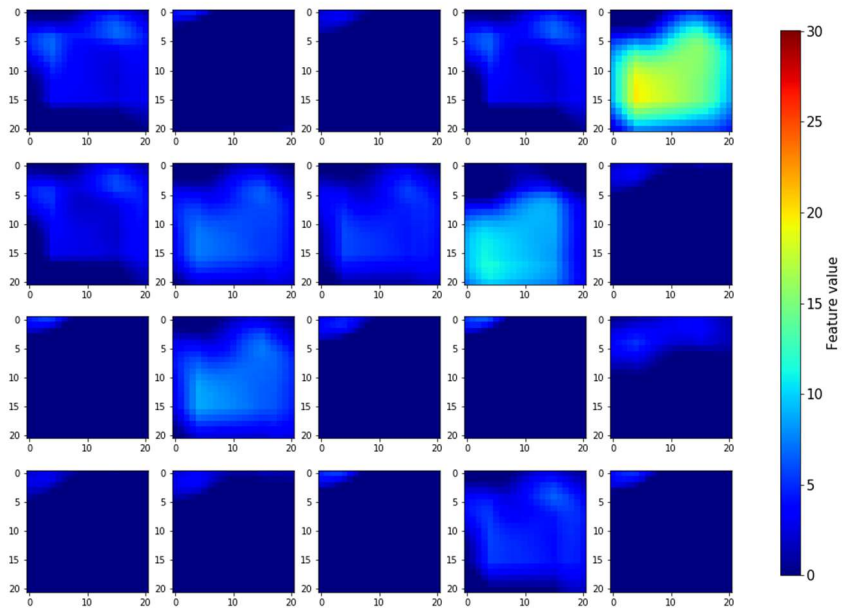


Figure D.11 The first layer of feature maps of the first convolutional layer at the channelized reservoir (input: pressure).

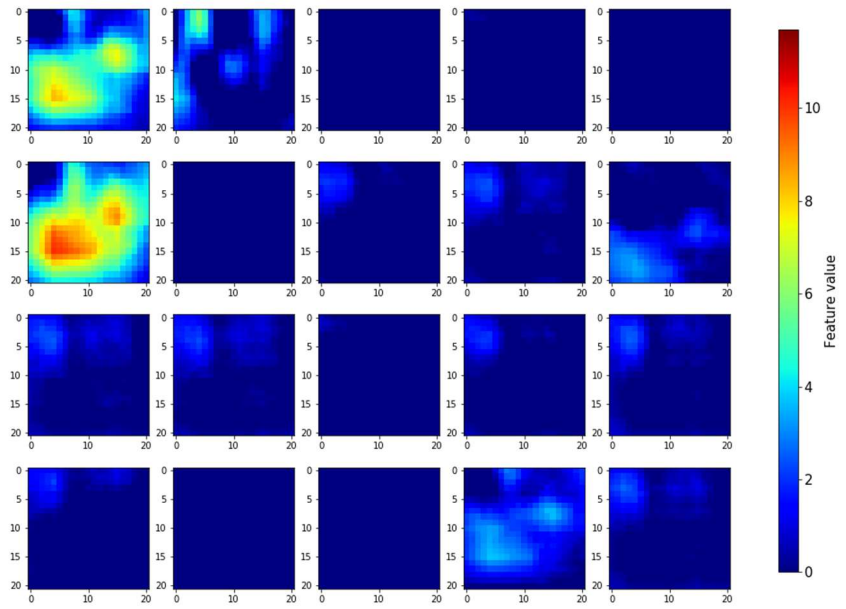


Figure D.12 The first layer of feature maps of the first convolutional layer at the channelized reservoir (input: oil saturation).

Appendix D. Keras source code: ANN for MNIST

MNIST(Modified National Institute of Standards and Technology) 데이터셋은 사람이 손으로 쓴 0부터 9까지의 숫자 이미지 70,000장을 디지털 형태로 저장한 데이터셋이다. 해당 데이터셋은 분류 문제를 위한 신경망 모델의 성능을 시험하는 용도로 널리 사용되고 있다. 개별 이미지의 크기는 28×28이며, 60,000장은 학습에 사용되고 10,000장은 시험에 사용된다. 이 코드에서는 60,000장의 학습 자료 중 12,000장을 검증 세트로 사용하였다.

```
from keras import layers, models

def ANN_models_func(Nin, Nh, Nout):
    x = layers.Input(shape=(Nin,))
    h = layers.Dense(Nh, activation='relu')(x)
    y = layers.Dense(Nout, activation='softmax')(h)
    main_model = models.Model(inputs=x, outputs=y)
    main_model.compile(loss='categorical_crossentropy', optimizer='adam',
metrics=['accuracy'])
    return main_model

from keras import datasets # mnist
from keras import utils # to_categorical

def Data_func():
    (X_train, y_train), (X_test, y_test) = datasets.mnist.load_data()
```

```
    Y_train = utils.to_categorical(y_train)
    Y_test = utils.to_categorical(y_test)

    img_num, img_row, img_col = X_train.shape
    X_train = X_train.reshape(-1, img_row*img_col)
    X_test    = X_test.reshape(-1, img_row*img_col)

    X_train = X_train/255.0
    X_test    = X_test/255.0

    return (X_train, Y_train), (X_test, Y_test)

import matplotlib.pyplot as plt

def plot_loss(history):
    plt.plot(history.history['loss'])
    plt.plot(history.history['val_loss'])
    plt.title('Model Loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend(['Train', 'Validation'], loc=0)

def plot_acc(history):
    plt.plot(history.history['acc'])
    plt.plot(history.history['val_acc'])
    plt.title('Model Accuracy')
    plt.xlabel('Epochs')
```

```
    plt.ylabel('Accuracy')
    plt.legend(['Train', 'Validation'], loc=0)

Nin = 784
Nh = 100
Nout = 10

main_model = ANN_models_func(Nin, Nh, Nout)
(X_train, Y_train), (X_test, Y_test) = Data_func()

history = main_model.fit(X_train, Y_train, epochs=15, batch_size=100, validation_split=0.2)
performance_test = main_model.evaluate(X_test, Y_test, batch_size=100)
print('Test Loss and Accuracy -> ', performance_test)

plot_loss(history)
plt.show()
plot_acc(history)
plt.show()

performance_train = main_model.evaluate(X_train[:48000], Y_train[:48000], batch_size=100)
performance_valid = main_model.evaluate(X_train[48000:], Y_train[48000:], batch_size=100)
print('Train Loss and Accuracy -> ', performance_train)
print('Validation Loss and Accuracy -> ', performance_valid)
```

Abstract

Optimization of Infill Well Placement Using Data-driven Multi-modal Convolutional Neural Network

Min-gon Chu
Department of Energy Systems Engineering
The Graduate School
Seoul National University

This study proposes a deep-neural-network based proxy that selects the optimal placement of infill well at a petroleum reservoir. A conventional artificial neural network (ANN) has a disadvantage in preserving features of spatial data that are static and dynamic petrophysical properties because all input data are forced to be transformed as a one-dimensional array and imported to the ANN. This study utilizes a convolutional neural network (CNN) to extract spatial features from multi-dimensional input arrays and evaluate the productivity of candidate infill wells as CNN outputs at affordable computation cost. CNN is trained to correlate between near-wellbore petrophysical properties and well productivity. Training dataset is acquired through reservoir simulation for selected infill well scenarios. Multi-modal learning is applied to the proxy for reflecting coupling effects from various types of petrophysical properties on well productivity.

The performance of the proposed multi-modal CNN is tested with application to shoreface and channelized reservoirs and compared to ANN results. Both reservoirs are inferred from the upper and lower parts of the SPE10 benchmark reservoir model, respectively. Overall, the CNN performance is superior to that of ANN for both reservoir models. For the channelized reservoir, the proxy outperforms the ANN as the discrete channel boundary helps the CNN distinguish productive sand channels from non-productive shale matrix. Top 20 prospects are selected using the multi-modal CNN, and

their productivities are validated in comparison with corresponding reservoir simulation results. The average relative errors are 1.50% and 4.27% for the shoreface and channelized reservoirs, respectively. The well location maximizing oil production among the validation results is considered the optimal solution.

Keywords: proxy, infill well placement, convolutional neural network, multi-modal learning
Student Number: 2011-21115