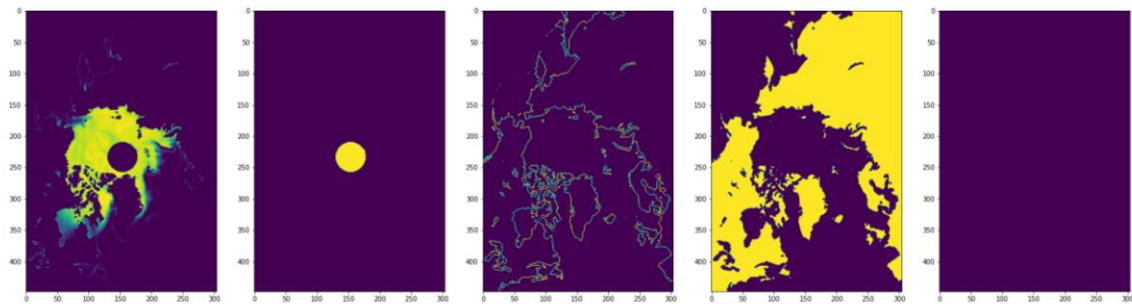


[배경] 지구 온난화 진행으로 지구 표면 온도 상승, 이상 기온, 해수면 상승 등 다양한 기후 변화가 관측 되고 있으며, 북극 해빙 또한 지구 온난화의 영향으로 매년 면적이 줄어들고 있는 중임

[목표] 과거 관측된 해빙 데이터를 통해 앞으로의 변화할 해빙 이미지 예측

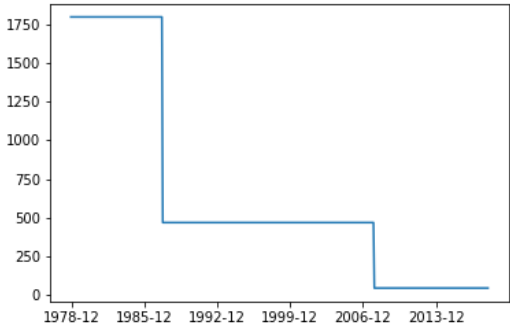
[데이터 EDA]

제공 데이터 셋 (이미지 데이터 (448 x 304))

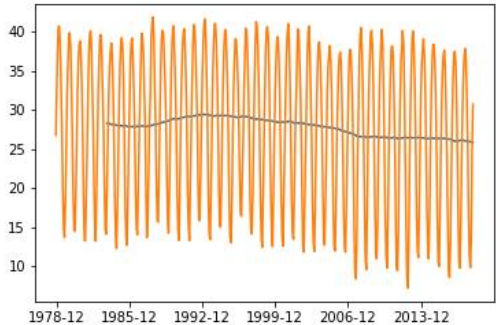


- 1978년 11월부터 2018년 12월까지 448 x 304 크기의 5개 월별 이미지 데이터를 제공
- 각 파일은 해빙 농도(0~250), 북극점(관측 불가 영역), 해안선, 육지선, 결측값을 제공
- 해안선,육지선, 결측값은 모든 데이터에서 동일하여 제외하고 사용

관측 불가 영역 추이



평균 해빙농도 추이(월별 vs 5년 이동평균)



- 관측 불가 영역이 기술 발전에 따라 줄어드는 것을 볼 수 있음
- 해빙농도의 월별 추이는 cyclical 하나 5년 이동평균으로는 조금씩 감소하는 것을 포착
- 감소 추이를 모델이 잘 잡을 수 있도록 최근 15년 데이터만을 사용

[Approach]

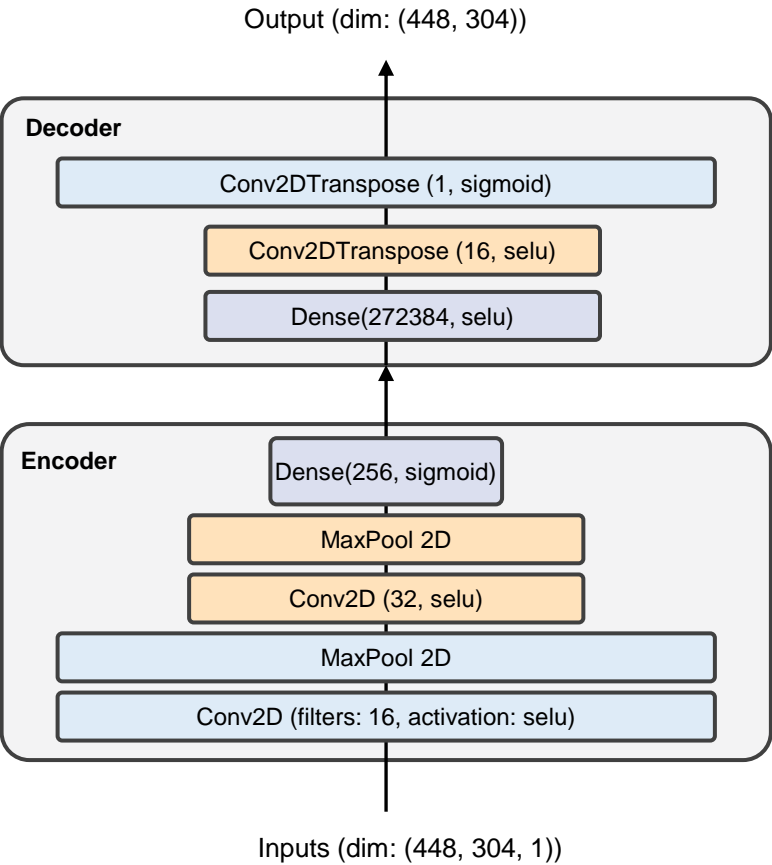
이미지의 dim이 448 x 304로 크기 때문에 computing burden을 줄이면서 학습/예측 할 수 있는 모델이 필요

1. Autoencoder의 encoder 를 통한 차원 축소 후, encoded 데이터를 이용하여 sequential 모델을 학습/예측
2. 원래 이미지를 여러 개의 작은 이미지로 cropping 하여 모델을 학습/예측 후 원래 이미지 크기로 재 가공

[Approach 1-1] AutoEncoder의 Encoder를 사용한 Dimension Reduction을 위한 AutoEncoder Model 탐색

- 예측이 불가했던 공간을 해당 이미지 해빙 농도의 Median 값으로 채운 후 448 x 304 크기의 이미지를 256 길이의 vector로 추출
- Conv2D를 사용한 여러 모델 중 Filters 16개와 32개의 Conv2D + MaxPool을 사용한 모델이 가장 좋은 결과를 보임
- SGD optimizer의 learning rate=0.1, momentum=0.9를 사용하여 700 epochs train 후 Val loss=0.0541 달성

[Best Model Architecture]

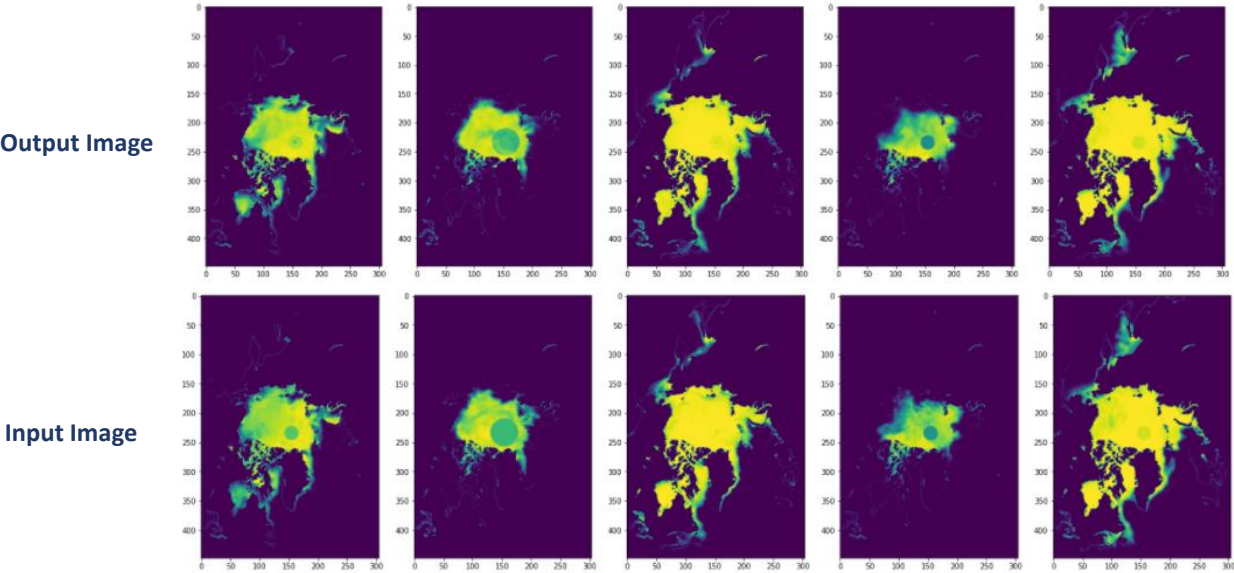


[Summary of Models]

Loss: Binary Crossentropy; optimizer : SGD

Model (All Ran for 100 epochs)	Validation Loss
One Conv2D(filters: 16)+Maxpool layer	0.0860
Two Conv2D(filters: 16 & 32)+Maxpool layers	0.0569
Two Conv2D(filters: 16 & 32)+Maxpool layers with Dense layer weights synced in Encoder and Decoder	0.0606
Two Conv2D layers with smaller filters (filters: 8 & 16)	0.0577
Three Conv2D layers (filters: 16 & 32 & 64)	0.0581

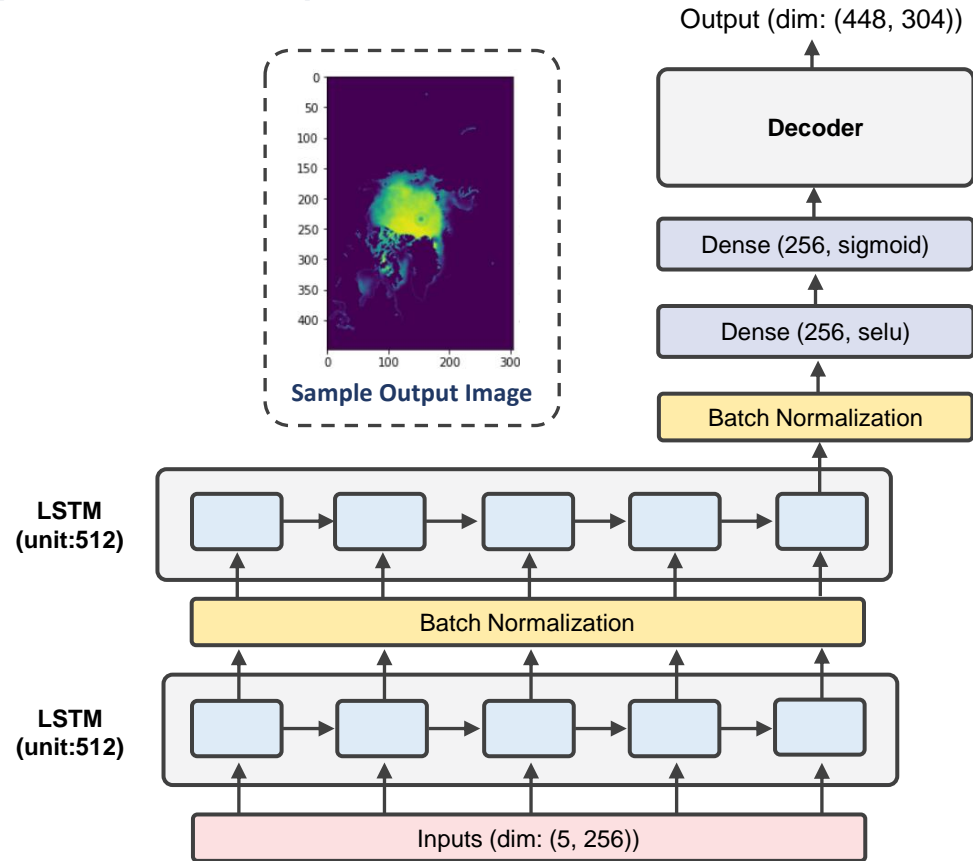
[Final AutoEncoder images(top), Inputted Raw Images(Bottom)]



[Approach 1-2] LSTM을 이용한 encoded sequential data 예측 모델 발굴

- **Input data** : predict year 전 5년간의 Encoded된 데이터; **Output Data** : predict year 이미지의 encoded data 예측 후 Decoder를 통한 image 생성
- Target image에 대한 loss를 정확히 파악할 수 있도록 Decoder 모델을 prediction 모델 후에 추가하여 이미지를 출력
- 2 개의 LSTM Layer과 Batch Norm을 더한 모델이 가장 좋은 결과를 보임. SGD(learning rate=0.001, momentum=0.999)와 400 epochs 훈련 후 Val Loss 0.0600 달성

[Best Model Architecture]



[Summary of Models]

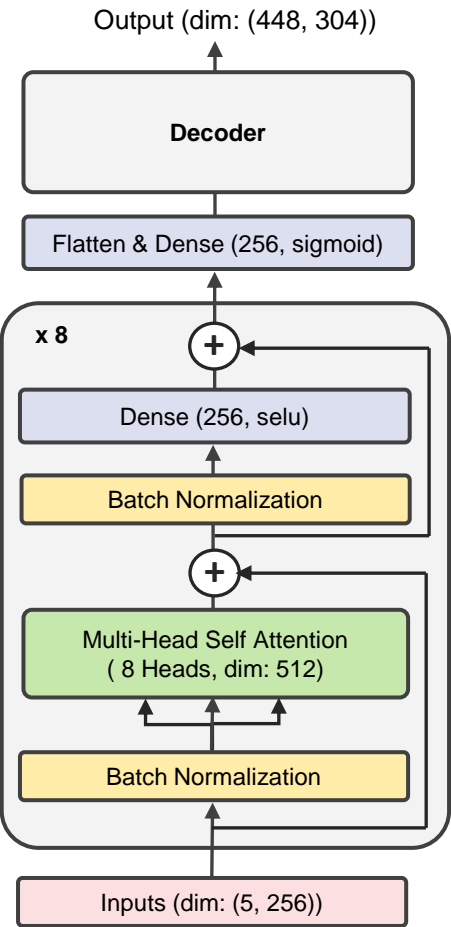
Model (All ran for 100 epochs)	Validation Loss
One LSTM layer (512 units)	0.0611
Two LSTM layers (512 units)	0.0614
Three LSTM layers (512 units)	0.0626
One LSTM layer with Batch Normalization (512 units)	0.0620
Two LSTM layer with Batch Normalization (512 units)	0.0605
Bigger LSTM unit with 1 layer (1024 units)	0.0611
Bigger LSTM unit with 2 layers (1024 units)	0.0615
Smaller LSTM unit with 1 layer (128 units)	0.0616
Conv2D before LSTM (2 filters, kernel size: 3, LSTM 512 units)	0.0609
Conv2D before 2 LSTM layers(2 filters, kernel size: 3, LSTM 512 units)	0.0622

Loss: Binary Crossentropy; optimizer : SGD

[Approach 1-2] Multi-Head Self Attention을 이용한 encoded sequential data 예측

- **Input data** : predict year 전 5년간의 Encoded된 데이터; **Output Data** : predict year의 encoded data 예측 후 Decoder를 통한 image 생성; **Decoder 추가**
- **Idea**: Sequential Data의 상호 유사도와 중요도를 파악하는 Self-Attention 모델을 사용해서 이미지 데이터 또한 학습 시도
- Google Research Team이 고안한 Transformer의 Encoder 모델과 Vision Transformer Model의 Encoder 모델을 인용
- BatchNorm, MultiHead, Dense를 이용한 모델에 Skip layer를 추가한 모델이 더 좋은 성능을 보임

[Best Model Architecture]



[Summary of Models]

Model(Ran for 100 epochs)	Validation Loss
(BatchNorm + MultiHead + BatchNorm + Dense) x 4	0.0635
((BatchNorm + MultiHead + BatchNorm + Dense) + Skip Layer) x 8	0.0611

Loss: Binary Crossentropy; optimizer : Adam

[참고 논문]

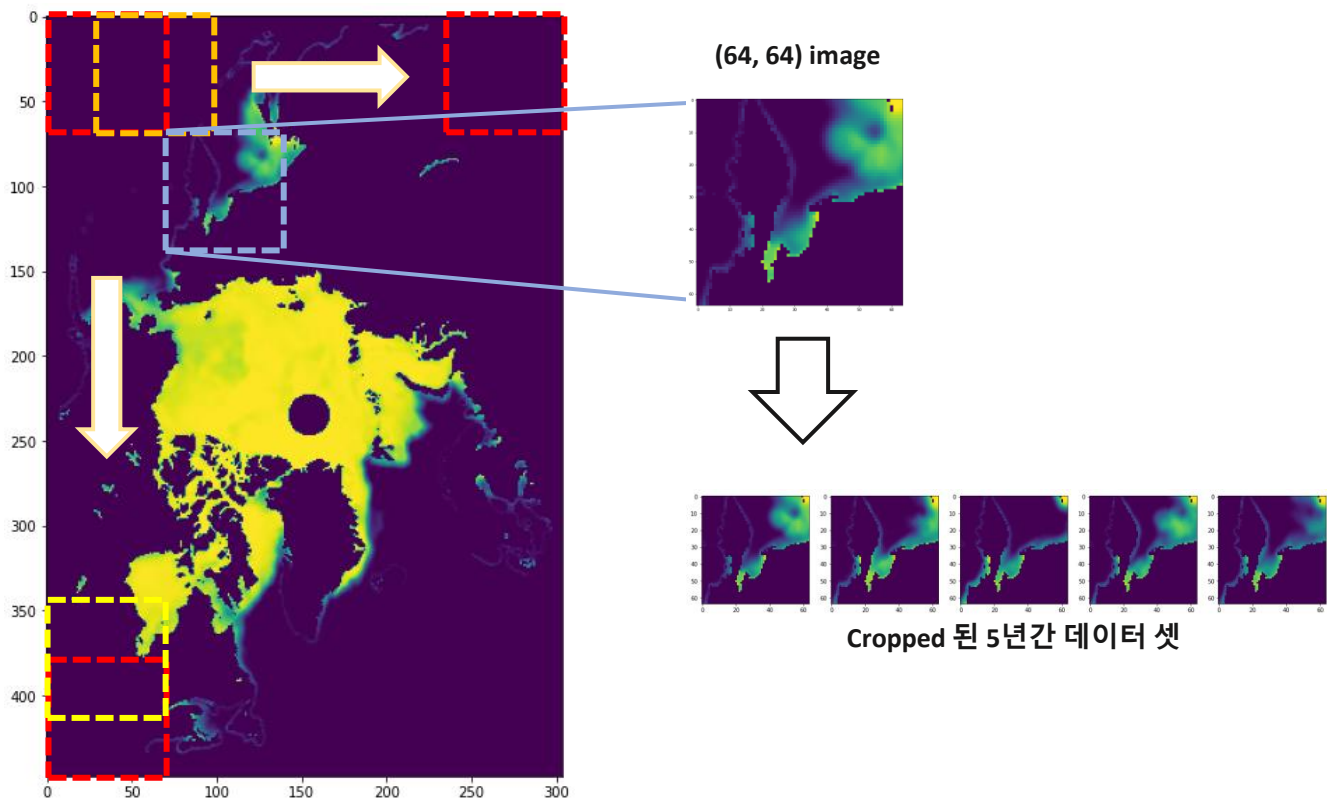
- An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2020, Google Research Team; [arXiv:2010.11929](https://arxiv.org/abs/2010.11929)
- Attention Is All You Need, 2017, Google Research Team; [arXiv:1706.03762](https://arxiv.org/abs/1706.03762)

[Approach 2-1] 분할 예측을 위해 이미지 Crop 및 예측 후 결합을 위한 Image pad 생성

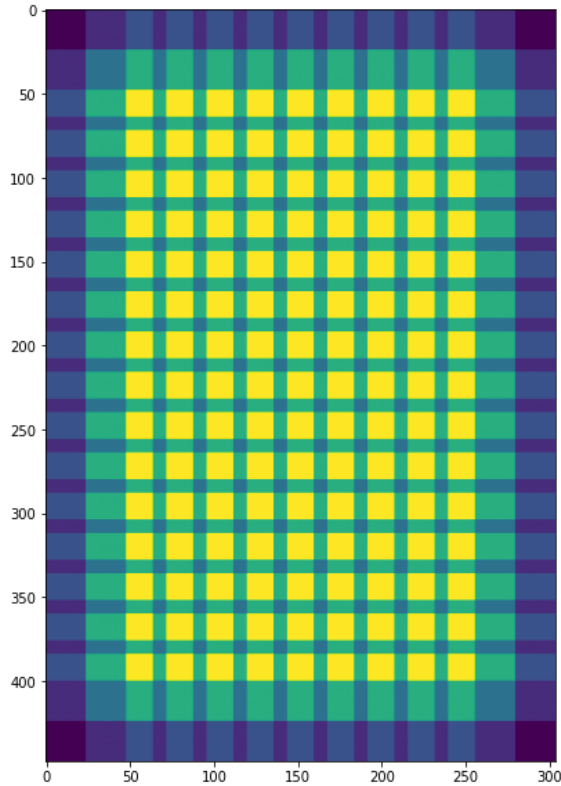
- 전체 이미지를 24 pixels 씩 아래쪽으로 움직이면서 64 x 64 사이즈 이미지로 cropping
- 예측을 위해 Cropping 된 이미지를 5년간 데이터 셋으로 re-grouping 한 후 input 데이터로 재구성
- 예측된 이미지의 pixel 값을 본래 이미지에 더하면 중첩 값들이 생김으로, 중첩 값들을 평균 내주기 위한 image pad를 생성
- Image pad는 24 pixel 씩 움직이면서 64 x 64 이미지 위치의 pixel 값에 1 씩 더해줌, 추후에 이를 element-wise division으로 이용

[Image Cropping 방식]

Window size = 64, strides = 24



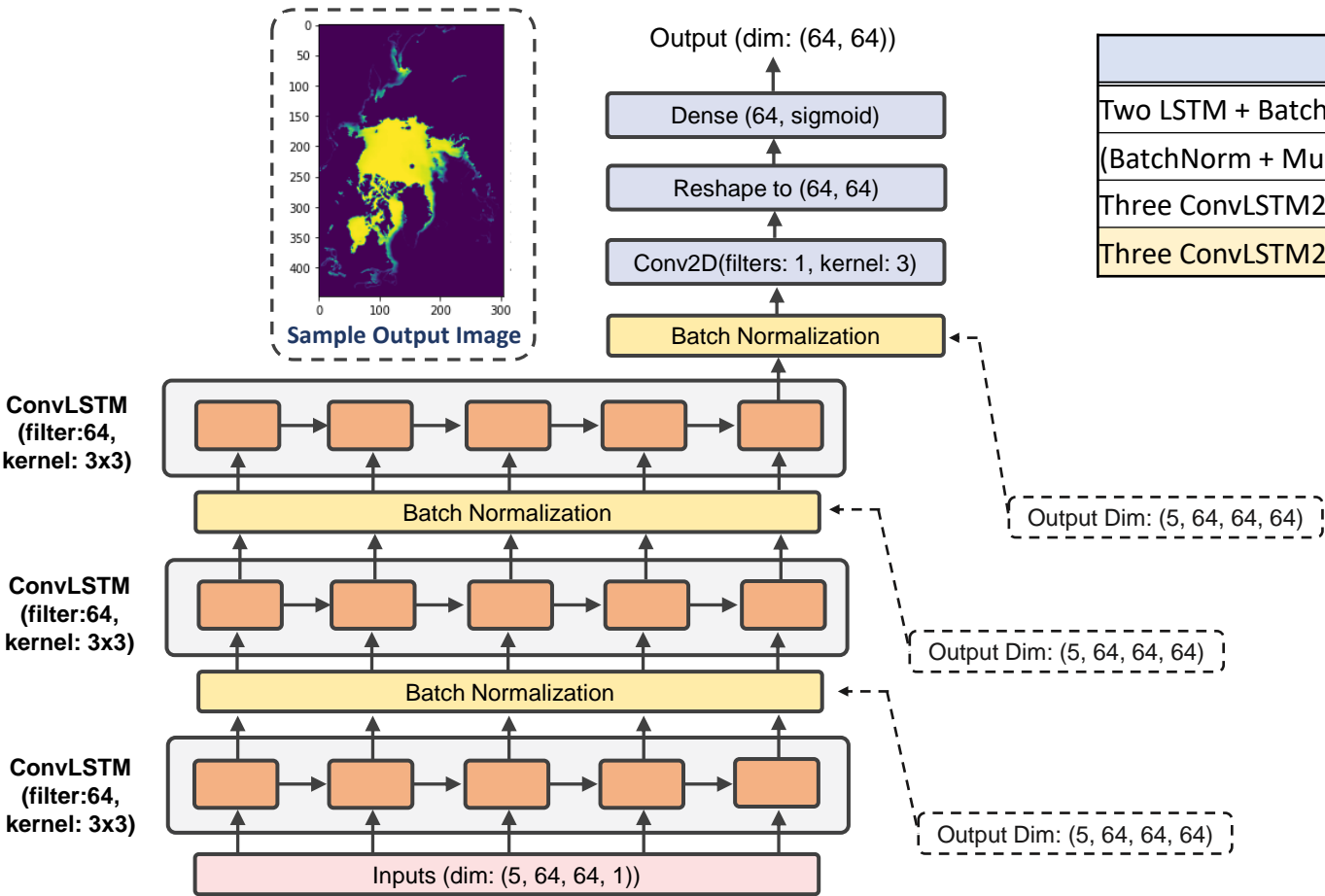
[전체 Image 예측시 Division에 사용되는 Image Pad]



[Approach 2-2] Cropped Image를 이용하여 큰 이미지를 분할하여 예측

- **Input data** : predict year 전 5년간의 cropped image; **Output Data** : predict year 이미지의 cropped된 한 부분
- 앞에서 보았던 모델들 중 좋은 결과를 보인 Two Layers LSTM+BatchNorm, Multi-Head Attention 모델과 시계열 이미지 처리가 가능한 ConvLSTM 이용
- ConvLSTM과 BatchNorm layer 3개와 sigmoid 함수를 last layer에 사용한 모델이 MAE 2.95를 보여주며 가장 좋은 성과를 보임
- 이미지가 Cropped 후에도 메모리가 크고 epoch train 시간이 오래걸려(1시간 이상) loss가 더이상 줄어들지 않는 모델들은 중간에 training을 멈추었음

[Best Model Architecture]



[Summary of Models]

Model	Validation Loss
Two LSTM + BatchNorm Layers	18.3465 (6 epochs)
(BatchNorm + MultiHead + BatchNorm + Dense) x 2 + Dense	19.5847 (11 epochs)
Three ConvLSTM2D + BatchNorm layers and Dense(64, relu)	5.1563 (20 epochs)
Three ConvLSTM2D + BatchNorm layers and Dense(64, sigmoid)	≈2.9500 (20 epochs)*

Loss: Mean Absolute Error; optimizer : Adam
* Model input and output scaled to values between 0 and 1.
So, MAE * 250 is used to approximate. (original mae:0.0118)

[Final Evaluation] Train한 모델들을 이용하여 Test Data를 예측

- 앞서 보았던 모델들을 이용하여 Test Data를 예측하고 pixel 값이 5~50인 pixel 값에 집중하여 Final Score 를 계산
- 각 모델을 epochs 중 Val Loss 가 가장 작은 값이 나오는 모델들로 저장하여 predict 하였음
- 세 모델의 Final Score은 0.1~0.2 정도 아주 큰 차이를 보이지는 않으나, Multi-Head Attention을 사용한 모델이 가장 좋은 결과 값을 보임

[Final Evaluation Metric]

```
import numpy as np

def mae_score(true, pred):
    # Regular MAE 계산
    score = np.mean(np.abs(true-pred))

    return score

def f1_score(true, pred):
    # target 이미지의 픽셀 값이 5~50 사이인 값만을 계산
    target = np.where((true>250*0.05)&(true<250*0.5))

    # 픽셀값 15를 threshold로 나눔
    true = true[target]
    pred = pred[target]
    true = np.where(true < 250*0.15, 0, 1)
    pred = np.where(pred < 250*0.15, 0, 1)

    # precision 과 recall 계산
    right = np.sum(true * pred == 1)
    precision = right / np.sum(true+1e-8)
    recall = right / np.sum(pred+1e-8)
    score = 2 * precision*recall/(precision+recall+1e-8)

    return score

def mae_over_f1(true, pred):
    # mae 를 f1 으로 나눔
    mae = mae_score(true, pred)
    f1 = f1_score(true, pred)
    score = mae/(f1+1e-8)

    return score
```

[Summary of Models]

Model		Final Score
With Autoencoder	Two Conv2D(filters: 16)+Maxpool layers	4.6961 (700 epochs)
With Autoencoder	((BatchNorm + MultiHead + BatchNorm + Dense) + Skip Layer) x 8	4.4870 (200 epochs)
With Cropped images	Three ConvLSTM2D + BatchNorm layers and Dense(64, sigmoid)	4.5689 (40 epochs)

[Predicted images from Jan. to Dec.]

