<div align="center">

|  |  |  |
| --- | --- | --- |
| **2019** | **Report – 2019/ 2월 2주** | **Seong-gu** |

</div>

**I. Plan for this week**

***Personal Research Subject***

* 미세먼지 예측 머신러닝 기반 모델링 최신 논문 계속 검토할 것.
* 바람의 영향에 의한 미세먼지 유입을 최적 고려한 기초 모델링 고안.

(locally-connected 2d lstm)

* ICIEE 학회 발표 PPT 제작 및 수정

**II. Plan for next week**

* locally-connected 2d lstm 모델 완성 및 기존 데이터로 테스트
* 학회 논문 발표 연습

**III. Response on Prof. Shin's feedback last week**

* 현재는 시간에 따른 예측이지만, 누락된 공간적인 데이터도 잘 예측하는 방법 검토. (Unsupervised learning이 될텐데 방법을 찾아보자.)
* 랜드마크 등 지형, 지역적 특성 수기 추가
* Feature를 잘 선택 (데이터 프로세싱으로 정확도 개선)
* 지역은 한국을 전범위로 하지 말고 적당한 수준으로 결정함.
* 시간대 설정
* 효율성 및 정확도 면에서 트레이드 오프 분석 설계 방안을 제기
* 온도, 습도, 대기압, 바람세기, 풍향 데이터 외에 날씨(눈/비/안개 등)도 포함하면 좋을듯함

**IV. Milestone**

* 시공간 데이터 보간 기법 활용 (forward + IDW)
* Locally-connected 2D LSTM 기초 모델 개발
* Locally-connected 2D LSTM 다층 구조 설계
* Attention 구조를 결합한 모델링도 고려.
* 연관관계가 검증된 추가 input feature (2차생성을 일으키는 SO2나 NOx) 활용

: forward-fixed 기법과 IDW 기법에 대해 가중치를

**V. Reports**

* 개인연구 진행사항
* **Locally Connected 2D LSTM layer 커스텀 제작**

Keras라이브러리에서 기본 제공하는 ConvLSTM 2D 레이어의 내부 구조를 변경하여, Locally Connected LSTM 2D 레이어를 제작하고 있으며, 다음주까지 변경을 완료하고 논문에서 사용한 데이터로 실제 성능 테스트를 진행할 예정입니다.

아래는 공개된 라이브러리(ConvLSTM, RNN, Conv 2D, Locally connected layer 등, 모델 오픈 소스) 출처입니다.

https://github.com/keras-team/keras/tree/master/keras/layers

관련된python class는 아래와 같습니다.

**class ConvRNN2D(RNN):**

RNN class를 상속받아 convolutional-recurrent layers의 기본 프레임을 정의합니다. time을 포함한 5D 입력

(samples, time, channels, rows, cols)을 받으며, return state나 sequence bool 변수가 true일 때 5D 출력(samples, timesteps, rows, cols, channels)을 합니다. False면, true 일 때, 4D 출력 (samples, new\_rows, new\_cols, filters)을 합니다.

<method>

compute\_output\_shape : Output shape를 정의합니다

build : input\_shape를 매개 변수로 받아, cell을 build하고, inner 변수(state\_spec, input\_spec)를 정의합니다.

get\_initial\_state : 매개변수로 규정된 초기상태를 정의하기 위해 inner 변수를 변경합니다.

reset\_states : stateful이라면 state를 reset합니다.

**class ConvLSTM2D(ConvRNN2D):**

ConvRNN2D를 상속받아 ConvLSTM 2D layer 의 기본 프레임을 정의합니다.

내부 Cell만 RNN Cell이 아닌 ConvLSTM2D Cell로 지정한다는 것 이외에 상속받은 ConvRNN2D의 프레임에서 특별히 변경된 사항은 없습니다.

**class ConvLSTM2DCell(Layer):**

Cell class는 time slice마다의 실제 내부 계산을 수행하는 단위블록입니다.

아래와 같이 call 함수에서 ConvLSTM 2d의 계산식을 수행합니다.

x\_i = self.input\_conv(inputs\_i, self.kernel\_i, self.bias\_i, padding=self.padding)

x\_f = self.input\_conv(inputs\_f, self.kernel\_f, self.bias\_f, padding=self.padding)

x\_c = self.input\_conv(inputs\_c, self.kernel\_c, self.bias\_c, padding=self.padding)

x\_o = self.input\_conv(inputs\_o, self.kernel\_o, self.bias\_o, padding=self.padding)

h\_i = self.recurrent\_conv(h\_tm1\_i, self.recurrent\_kernel\_i)

h\_f = self.recurrent\_conv(h\_tm1\_f, self.recurrent\_kernel\_f)

h\_c = self.recurrent\_conv(h\_tm1\_c, self.recurrent\_kernel\_c)

h\_o = self.recurrent\_conv(h\_tm1\_o, self.recurrent\_kernel\_o)

i = self.recurrent\_activation(x\_i + h\_i)

f = self.recurrent\_activation(x\_f + h\_f)

c = f \* c\_tm1 + i \* self.activation(x\_c + h\_c)

o = self.recurrent\_activation(x\_o + h\_o)

h = o \* self.activation(c)

그리고 아래와 같이 kernel 차원이 정의됩니다. (input\_dim은 input channel을 의미합니다.)

kernel\_shape = self.kernel\_size + (input\_dim, self.filters \* 4)

즉, 2D의 경우 kernel 차원은 4차원인 (kernel의 row, kernel의 col, input channel, filter \*4)로 되며,

ConvLSTM에는 3개의 게이트와 memory sell이 포함되어 2D Conv에 비해 filter가 4배 많은 것을 제외하고 동일합니다.

**반면, Locally-connected 의 기본 구조는**

kernel\_shape = (output\_row \* output\_col , input\_filter \*self.kernel\_size[0] \* self.kernel\_size[1], self.filters )

로 3차원이 됩니다.

곧, ConvLSTM와 Locally-connected 기본 구조에는 kernel 차원의 차이가 있으며 이를 해결하는 편집작업 및 디버깅 작업을 진행 중에 있습니다.