The University of Texas at Austin

CNN-RNN FORWARD PROXY MODELING FOR CO2 MONITORING

Team MOMA: Misael Morales & Oriyomi Raheem

GEO 391 – Machine Learning Applications in Geoscience

Spring 2022

Contents

PROBLEM STATEMENT

Question, Problem, and Proposal

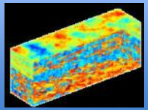
Problem Statement

- Reservoir simulation is crucial for subsurface energy resource engineering
  - Often, it is very complex and time-consuming
- Develop a deep learning framework for forward reservoir simulation
  - Better computational efficiency
  - Accuracy trade-off
- Exploit latent space dynamics for timelapse predictions using CNN-RNN architecture

Problem Statement

Reservoir Model

(SPE10)



$(220 \times 60 \times 85) \rightarrow 85 \ @ \ (60 \times 60)$
1 injector @ $(30,30)$
CO_2 @ $5 \ m^3/day$
5 years injection, monitor monthly
Tarbert (Gaussian) + Ness (fluvial)

Data Processing

(Python)

Reshape to images
Data augmentation (rotation)
Random Shuffling
Min-Max Normalization
Train/Test split

Numerical Simulation

(MRST)

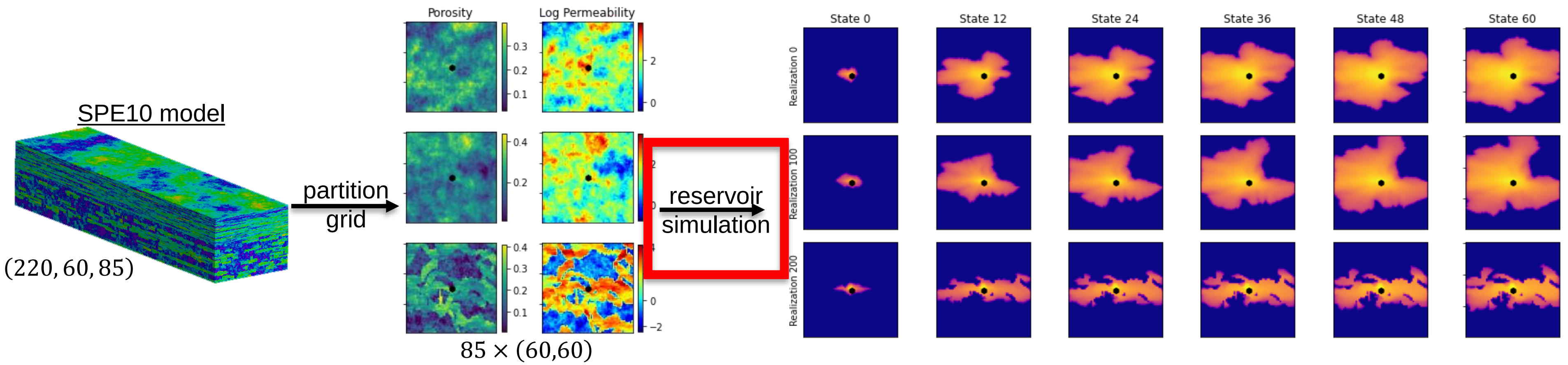
Two-phase water-gas model
FD + Automatic Differentiation
Output: dynamic pressure & saturation fields
$(255, 3600)$ & $(255, 3600, 60)$

Deep Learning

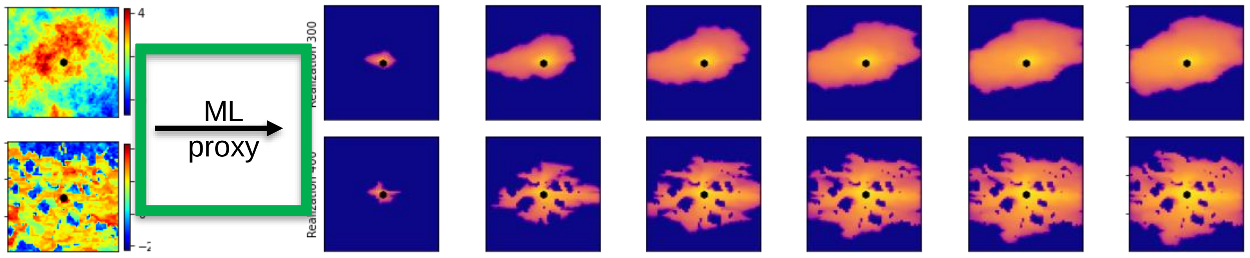
(Keras)

Encoder: Conv2D
Recurrent: GRU
Decoder: Conv3DTranspose
Compile, Fit & Predict

Problem Statement



CNN-RNN proxy to exploit latent space dynamics



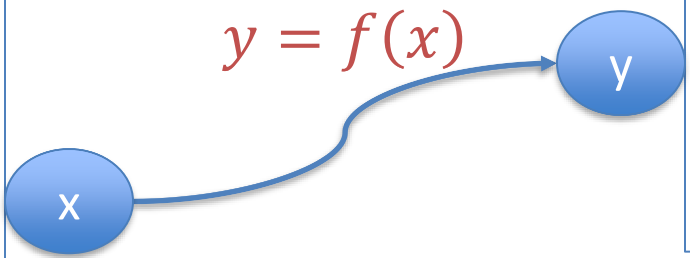
RESERVOIR SIMULATION

Model construction and computation

Reservoir Simulation

Inputs

- Grid
- Rock properties
- Fluid properties
- Initial state
- Wells
- Boundary conditions
- Schedule
- Solver



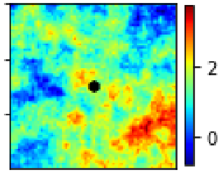
Outputs

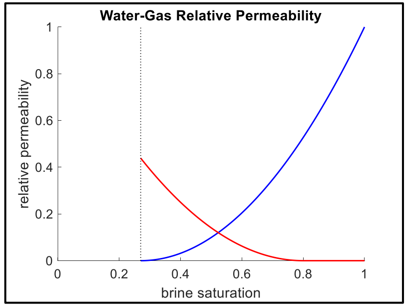
- Numerical report
- Well solution
- Pressure states
- Saturation states

$$\vec{\nabla}\cdot\left(\rho_\alpha \vec{u}_\alpha\right) + \tilde{q}_{m,\alpha} = -\frac{\partial(\varphi S_\alpha \rho_\alpha)}{\partial t}, \qquad \tilde{q}_{m,\alpha} = \frac{q_{m,\alpha}}{V_b},$$

$$\vec{u}_\alpha(x) = -\frac{k_{r,\alpha}K}{\eta_\alpha}\left(\vec{\nabla}p_\alpha - \rho_\alpha g \vec{\nabla}D\right),$$

Reservoir Simulation





- High-fidelity simulations are performed using MRST
- 255, 2D realizations with 1 injector
  - Initially water saturated
  - CO2 injection @ 5 m^3/day
  - 5 years, monitored monthly
  - Automatic Differentiation framework

- Parallelized over 10 cores on an Intel i9-10900K @ 5000 MHz
  - ≈ 20 seconds per realization

```
%% Generate Models & Run Simulation
N = size(all_poro,2);    %number of realizations (255)
M = size(total_time,1); %number of schedule timesteps (60)

parfor i=1:N
    fprintf('Simulation %i\n', i)
    rock                          = gen_rock(all_poro, all_perm, i)
    W                             = gen_wells(G, rock)
    [schedule, dT1]               = gen_schedule(W, bc, timestep1)
    [model, wellSol, states] = gen_simulation(G, rock, fluid, initState, schedule)
    result{i} = states;
end
```

DATA PROCESSING

Preparing data for deep learning

Data Processing

1. HFS results are sliced to: $x \Longrightarrow [poro, perm]$ and $y \Longrightarrow [saturation, pressure]$
   - Exported as MATLAB (*$.m$*) files
   - Imported using *SciPy*

```
Porosity shape: (255, 3600) | Permeability shape: (255, 3600)
Pressure shape: (255, 3600, 60) | Saturation shape: (255, 3600, 60)
```

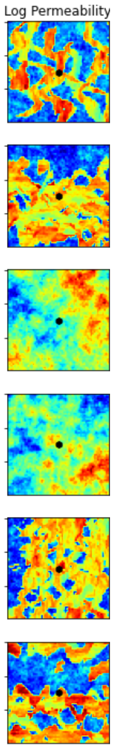
```
Porosity shape: (255, 60, 60)     | Permeability shape: (255, 60, 60)
Pressure shape: (255, 60, 60, 60) | Saturations  shape: (255, 60, 60, 60)
```

2. Reshape to 2D images and 3D “videos”

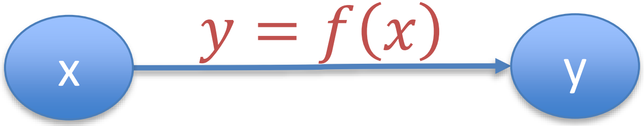
3. Data augmentation by 90° rotation

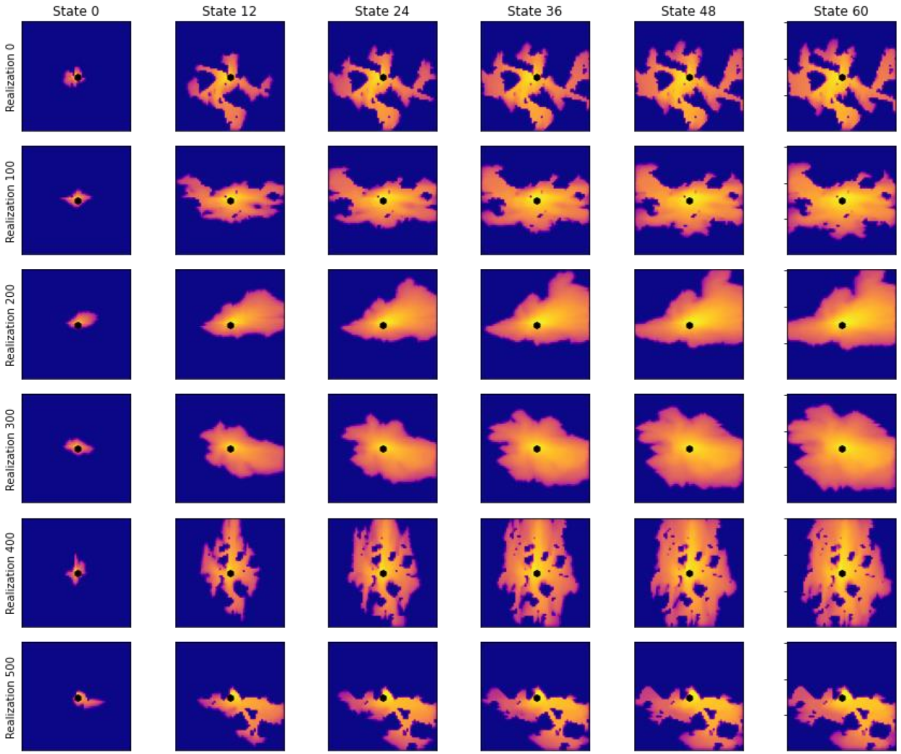
4. Shuffle concatenated dataset
   - Make proxy agnostic to orientation, learn true flow physics

Data Processing



$y = f(x)$





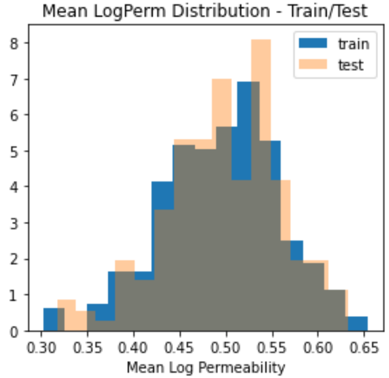
Data Processing

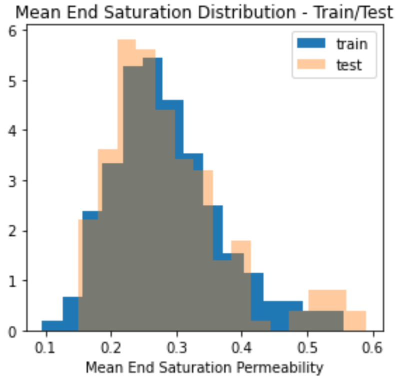
5. Min-Max Normalization
   – For each realization & for each state:

$$\hat{y} = \frac{y - y_{\min}}{y_{max} - y_{min}}$$

6. Train/Test split
   – Randomly assigned train/test index

```
X_train shape: (340, 60, 60, 1) | y_train shape: (340, 60, 60, 60, 1)
X_test  shape: (170, 60, 60, 1) | y_test  shape: (170, 60, 60, 60, 1)
```



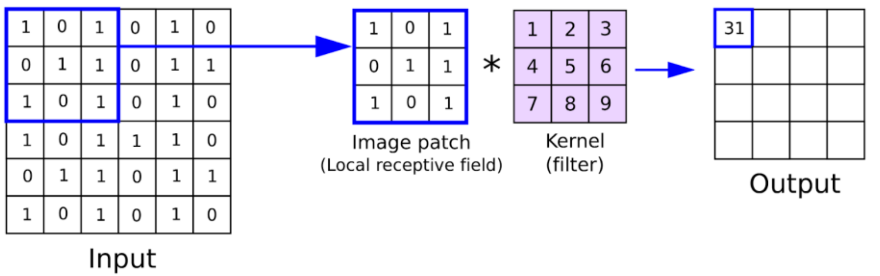


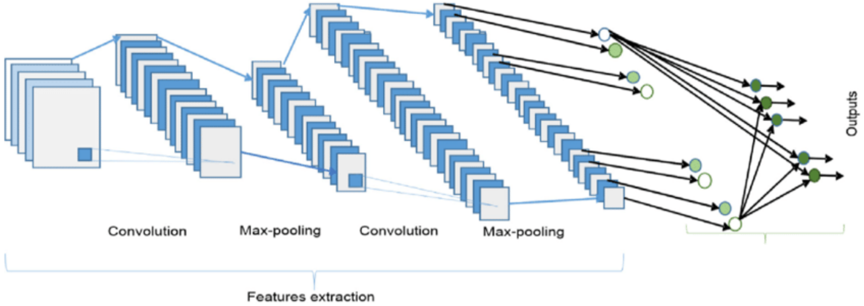
CNN-RNN PROXY MODEL

Convolution & Recurrent layers, Latent space representations, and Model building

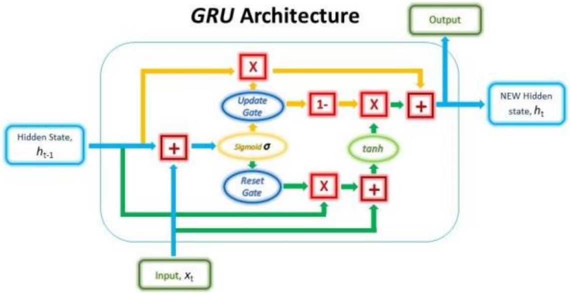
CNN-RNN Proxy Model

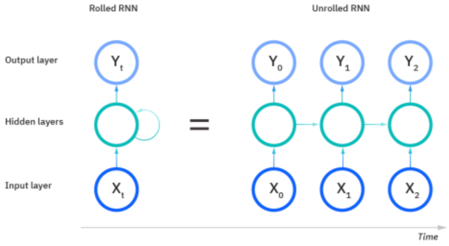
The convolutional layer



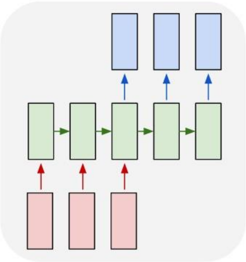


The recurrent layer



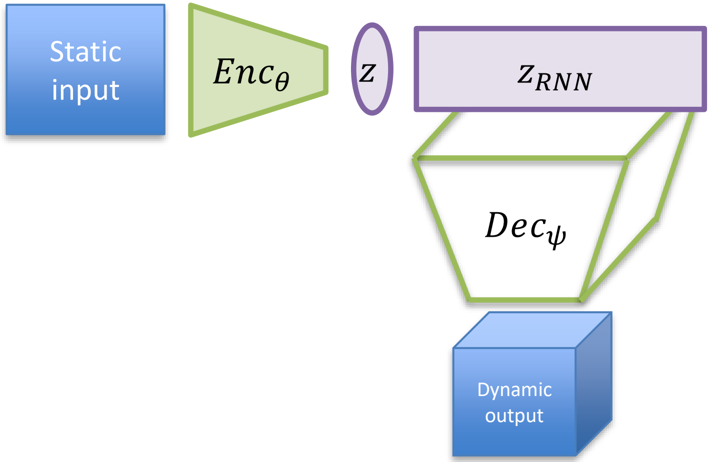


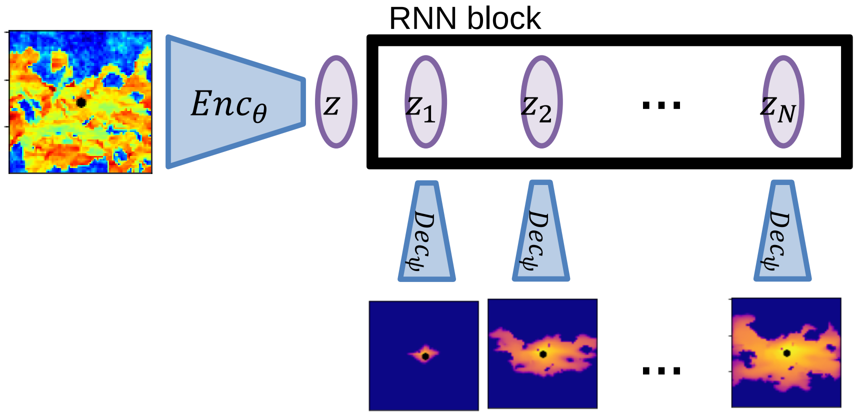
many to many



Return sequence = *True*

CNN-RNN Proxy Model

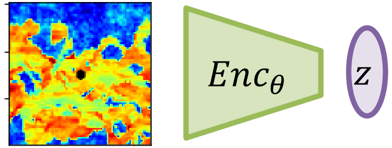




Total # of Parameters: 930,121

CNN-RNN Proxy Model

The Encoder



$(None, 60, 60, 1)$

Conv Block 1 — $N_f = 8$

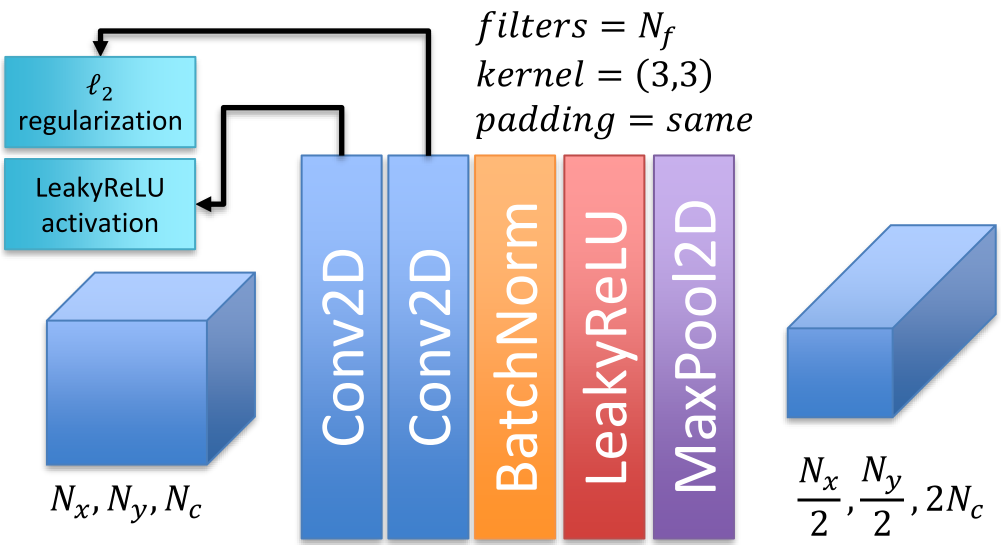
Conv Block 2 — $N_f = 16$

Conv Block 3 — $N_f = 32$

Conv Block 4 — $N_f = 64$

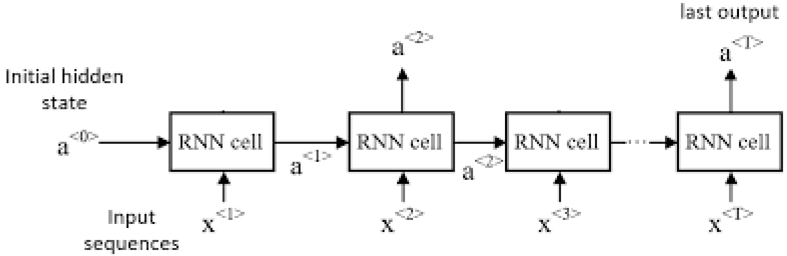
$(None, 3, 3, 64)$

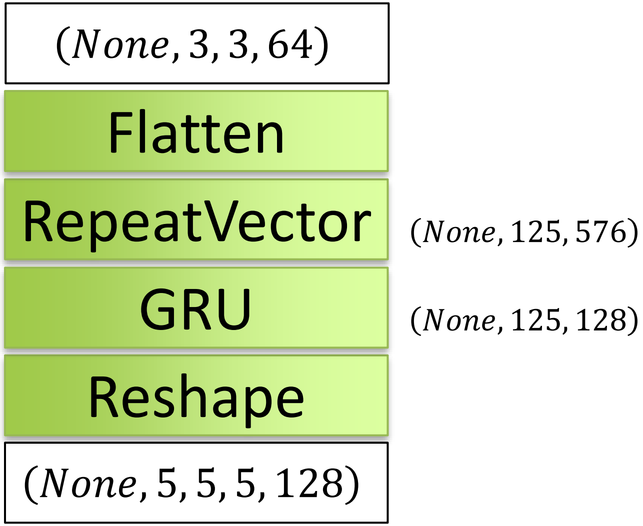
Convolutional block



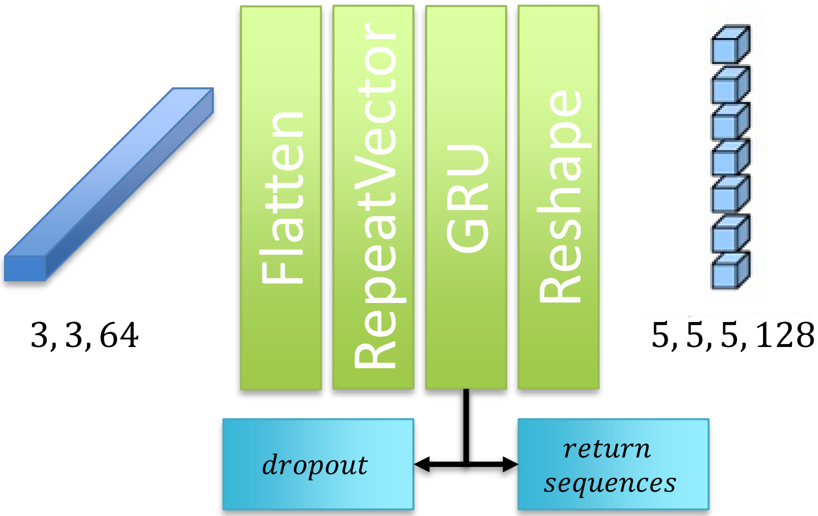
CNN-RNN Proxy Model

The Recurrent block



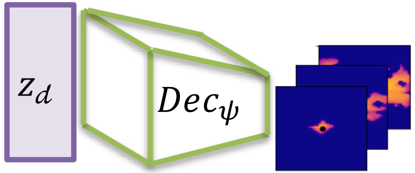


Recurrent block



CNN-RNN Proxy Model

The Decoder



$(None, 5, 5, 5, 128)$

ConvT Block 1 — $N_f = 64, stride = 1$

ConvT Block 2 — $N_f = 32, stride = N_s$

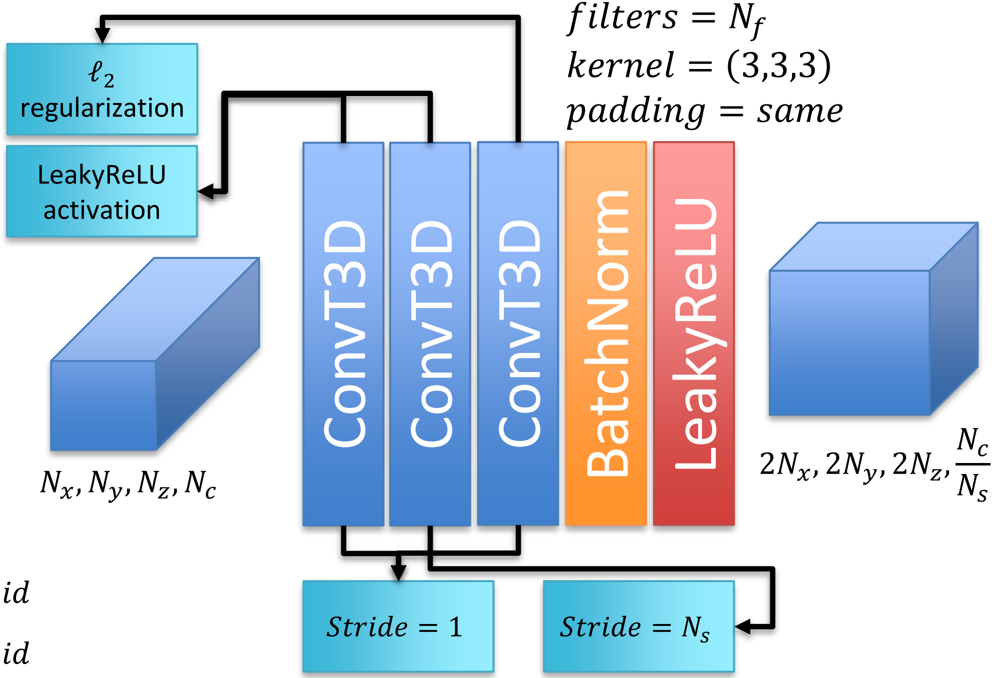
ConvT Block 3 — $N_f = 16, stride = 1$

Output Block
- ConvT3D — $N_f = 8, sigmoid$
- Conv3D — $N_f = 1, sigmoid$

$(None, 60, 60, 60, 1)$

Transpose Convolutional block

$filters = N_f$
$kernel = (3,3,3)$
$padding = same$



TRAINING & TESTING

Performance and Visualization

Training & Testing

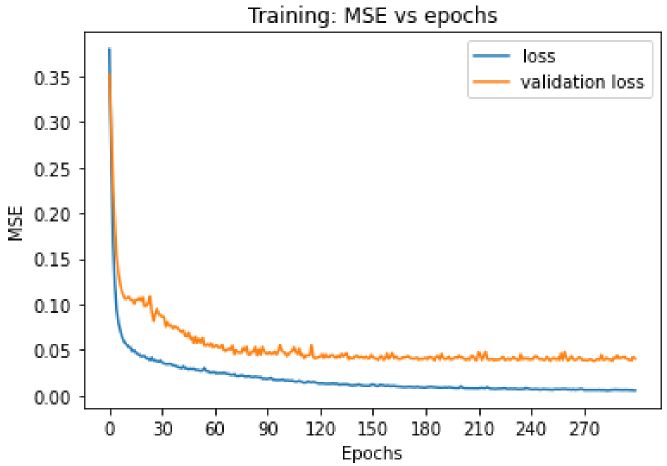
$$\mathbf{g}_t \leftarrow \nabla_{\theta_{t-1}} f_t(\theta_{t-1})$$

$$\mathbf{m}_t \leftarrow \mu_t \mathbf{m}_{t-1} + \alpha_t \mathbf{g}_t$$

$$\theta_t \leftarrow \theta_{t-1} - (\mu_{t+1}\mathbf{m}_t + \alpha_t \mathbf{g}_t)$$

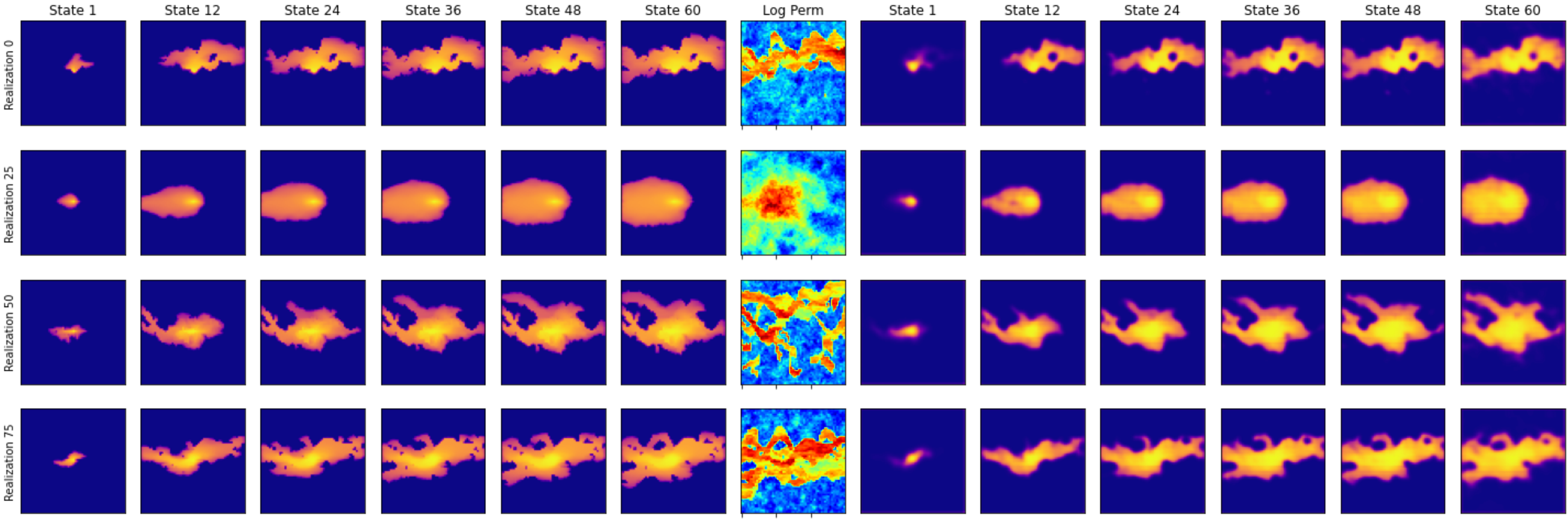
$$\theta_t \leftarrow \theta_{t-1} - \alpha_t \left(\frac{\mu_{t+1}\mathbf{m}_t}{1 - \prod_{i=1}^{t+1} \mu_i} + \frac{(1-\mu_t)\mathbf{g}_t}{1 - \prod_{i=1}^{t} \mu_i} \right)$$

- Compile
  - Optimizer = Nadam (Adam with Nesterov momentum)
  - Loss = MSE
- Fit
  - Epochs = 300
  - Batch size = 40
  - Validation split = 0.25
  - Workers = 10
- ≈ 17 minutes on Nvidia RTX 3080



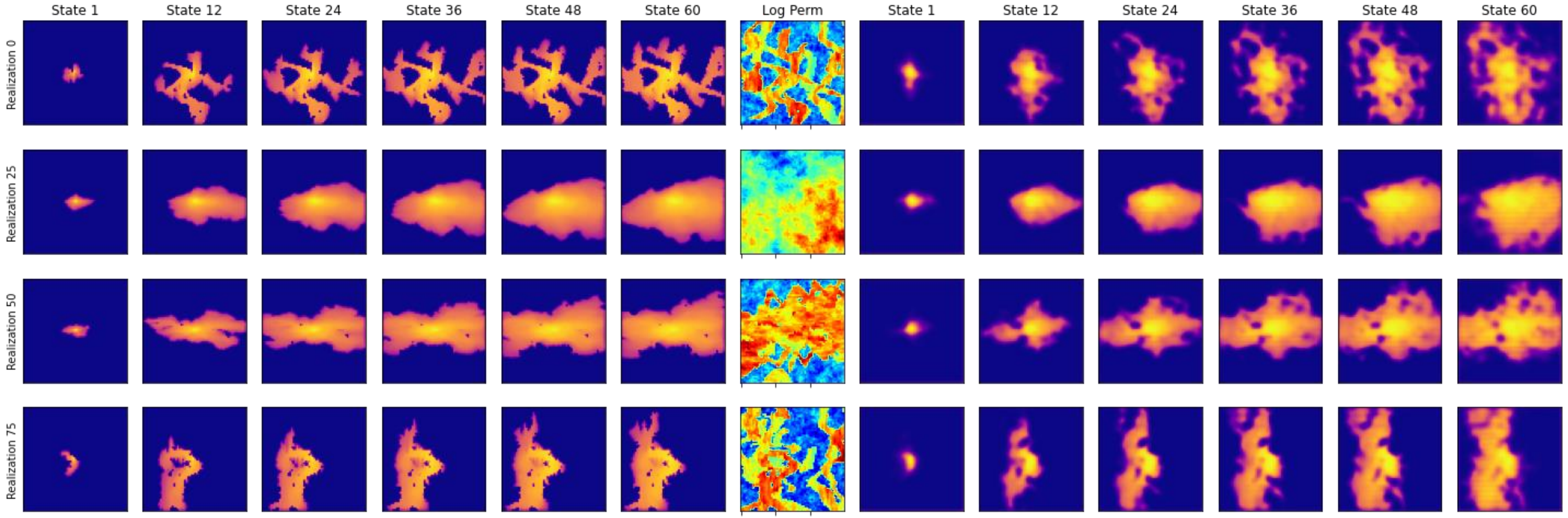
Training & Testing

- Use trained model to show training predictions
  - MSE = 0.016 ; SSIM = 0.817



Training & Testing

- Testing predictions
  - MSE = 0.040 ; SSIM = 0.563



RESULTS, DISCUSSION & CONCLUSIONS

Lessons Learned, Possible Applications, and Future Directions

Results, Discussion & Conclusions

- High-Fidelity Simulations
  - Crucial yet costly step in reservoir characterization and forecasting
  - HFS performed in MRST from SPE10 partition
  - Approximately 20 seconds per realizations

- Data Processing
  - Data: feature (static – permeability) & target (dynamic – saturation)
  - Augmentation by rotation; Random shuffling; Min-Max Normalization; Train/Test split
  - Optimized for use in Keras deep learning framework to make proxy model learn the physics of the system

Results, Discussion & Conclusions

- Proxy Model
  - Block architecture: Encoder – Recurrent – Decoder
  - Approximately 930,000 parameters
  - Each prediction is ≈ 0.5 milliseconds $\Longrightarrow 40{,}000$x speedup!

- Results
  - The proxy model is extremely efficient in predicting dynamic saturation states from a static permeability map.
  - Train MSE, SSIM = 0.016, 0.817
  - Test MSE, SSIM = 0.040, 0.563

Results, Discussion & Conclusions

- Conclusion
  - Including a SSIM loss function to improve testing predictions
  - Expand training data through augmentation or generate more realizations
  - Applications:
    - Uncertainty quantification
    - History matching / parameter estimation / model calibration
    - Closed-loop optimization
  - Other areas:
    - Groundwater flows
    - Contaminant transport
    - Petroleum production

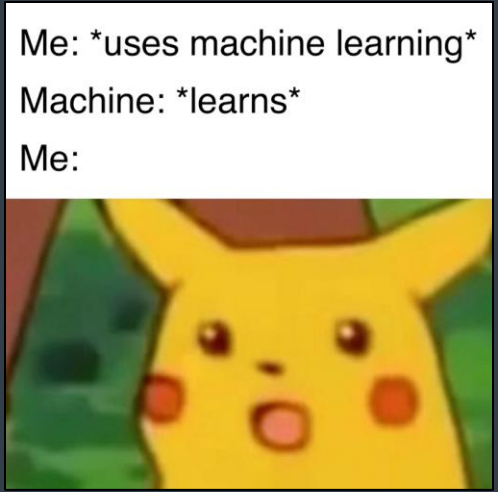
References

1. Maldonado-Cruz, Eduardo, and Michael J Pyrcz. (2022) “Fast Evaluation of Pressure and Saturation Predictions with a Deep Learning Surrogate Flow Model.” Journal of petroleum science & engineering 212
2. Kim, Y. D., & Durlofsky, L. J. (2022). "Convolutional-Recurrent Neural Network Proxy for Robust Optimization and Closed-Loop Reservoir Management." arXiv preprint arXiv:2203.07524.
3. Kaur, Harpreet et al. (2022) “Time-Lapse Seismic Data Inversion for Estimating Reservoir Parameters Using Deep Learning.” Interpretation (Tulsa) 10.1
4. S. Pan, S.L. Brunton, and J.N. Kutz (2022) "Neural Implicit Flow: a mesh-agnostic dimensionality reduction paradigm of spatio-temporal data." arXiv preprint arXiv:2204.03216
5. Joon, Shams, Dawuda, Ismael, Morgan, Eugene, and Sanjay Srinivasan. (2022) "Rock Physics-Based Data Assimilation of Integrated Continuous Active-Source Seismic and Pressure Monitoring Data during Geological Carbon Storage." SPE Journal.
6. Gonzalez, Keyla, and Siddharth Misra. (2022) “Unsupervised Learning Monitors the Carbon-Dioxide Plume in the Subsurface Carbon Storage Reservoir.” Expert systems with applications 201
7. K.G. Gurjao, E. Gildin, R. Gibson, and M. Everett. (2022) "Estimation of Far-Field Fiber Optics Distributed Acoustic Sensing DAS Response Using Spatio-Temporal Machine Learning Schemes and Improvement of Hydraulic Fracture Geometric Characterization." SPE Hydraulic Fracturing Technology Conference and Exhibition, USA
8. Salazar, Jose J et al. (2022) “Fair Train-Test Split in Machine Learning: Mitigating Spatial Autocorrelation for Improved Prediction Accuracy.” Journal of petroleum science & engineering 209:109885–.
9. Tang, Meng, Yimin Liu, and Louis J Durlofsky. (2021) “Deep-Learning-Based Surrogate Flow Modeling and Geological Parameterization for Data Assimilation in 3D Subsurface Flow.” Computer methods in applied mechanics and engineering 376:113636–.
10. H. Jo, Y. Cho, M.J. Pyrcz, H. Tang, and P. Fu (2021) "Machine learning-based porosity estimation from spectral decomposed siesmic data." arXiv preprint arXiv:2111.13581
11. Wen, Gege, Catherine Hay, and Sally M Benson. (2021) “CCSNet: A Deep Learning Modeling Suite for CO2 Storage.” Advances in water resources 155:104009–.
12. Alsulaimani, Thamer , and Mary Wheeler. (2021) "Reduced-Order Modeling for Multiphase Flow Using a Physics-Based Deep Learning." SPE Reservoir Simulation Conference
13. E.J.R. Coutinho, M.J. Aqua and E. Gildin. (2021) "Physics-Aware Deep-Learning-Based Proxy Reservoir Simulation Model Equipped with State and Well Output Prediction." SPE Reservoir Simulation Conference, Virtual.
14. Ciriello, V., Lee, J. & Tartakovsky, D.M. (2021) "Advances in uncertainty quantification for water resources applications." Stoch Environ Res Risk Assess 35, 955–957
15. Pan, W., Torres-Verdín, C. & Pyrcz, M.J. (2021) "Stochastic Pix2pix: A New Machine Learning Method for Geophysical and Well Conditioning of Rule-Based Channel Reservoir Models." Nat Resour Res 30, 1319–1345
16. Wu, Hao et al. (2021) “A Multi-Dimensional Parametric Study of Variability in Multi-Phase Flow Dynamics During Geologic CO2 Sequestration Accelerated with Machine Learning.” Applied energy 287:116580–.
17. Santos, J.E., Yin, Y., Jo, H. et al. (2021) "Computationally Efficient Multiscale Neural Networks Applied to Fluid Flow in Complex 3D Porous Media." Transp Porous Med 140, 241–272
18. Chan, S., Elsheikh, A.H. (2020) "Data-driven acceleration of multiscale methods for uncertainty quantification: application in transient multiphase flow in porous media." Int J Geomath 11,3
19. Cheung, S.W., Chung, E.T., Efendiev, Y. et al. (2020) "Deep global model reduction learning in porous media flow simulation." Comput Geosci 24, 261–274
20. Almasov, Azad , Onur, Mustafa , and Albert C. Reynolds. (2020) "Production Optimization of the CO2 Huff-N-Puff Process in an Unconventional Reservoir Using a Machine Learning Based Proxy." SPE Improved Oil Recovery Conference, Virtual
21. Jiang, Chiyu lmaxr et al. (2020) “MESHFREEFLOWNET: A Physics-Constrained Deep Continuous Space-Time Super-Resolution Framework.” SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. IEEE 1–15.
22. J. Nagoor Kani, Elsheikh, A.H. (2019) "Reduced-Order Modeling of Subsurface Multi-phase Flow Models Using Deep Residual Recurrent Neural Networks." Transp Porous Med 126, 713–741
23. Jayne, Richard S, Hao Wu, and Ryan M Pollyea. (2019) “Geologic CO2 Sequestration and Permeability Uncertainty in a Highly Heterogeneous Reservoir.” International journal of greenhouse gas control 83.C:128–139.
24. K.-A. Lie. (2019) "An Introduction to Reservoir Simulation Using MATLAB/GNU Octave: User Guide for the MATLAB Reservoir Simulation Toolbox (MRST)." Cambridge University Press
25. Brunton, Steven L., and Jose Nathan Kutz. (2019) "Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control." 1st ed. Cambridge University Press
26. Guo, Zhenyu, and Albert C Reynolds. (2018) “Robust Life-Cycle Production Optimization With a Support-Vector-Regression Proxy.” SPE journal 23.6:2409–2427
27. Naraghi, Morteza Elahi, Spikes, Kyle , and Sanjay Srinivasan. (2017) "3D Reconstruction of Porous Media From a 2D Section and Comparisons of Transport and Elastic Properties." SPE Res Eval & Eng 20:342–352
28. Ampomah, W et al. (2017) “Optimum Design of CO2 Storage and Oil Recovery Under Geological Uncertainty.” Applied energy 195
29. J. Nagoor Kani, Elsheikh, A.H. (2017) "DR-RNN: A deep residual recurrent neural network for model reduction." arXiv preprint arXiv:1709.00939

Thank you!

Questions?

Link to notebook:
https://github.com/misaelmmorales/CNN-RNN-CO2



Misael Morales:
PhD Student in Petroleum & Geosystems Engineering
Advisors: Dr. Michael Pyrcz, Dr. Carlos Torres-Verdin
GitHub | Linkedin | Website |
misaelmorales@utexas.edu

Oriyomi Raheem:
PhD Student in Petroleum & Geosystems Engineering
Advisor: Dr. Carlos Torres-Verdin
oriyomiraheem@utexas.edu

BACKUP SLIDES

Simulation Variables

- Reference pressure: 30 mega Pascals
- Reference temperature: $94℃$
- Water compressibility: 0
- Rock compressibility: 4.35×10^{-5} bars
- Water viscosity: 8×10^{-4} Pascal-second
- CO2 viscosity: 5.68×10^{-5} Pascal-second
- Residual water saturation: 0.27
- Residual CO2 saturation: 0.20
- Water viscosity: 1000

Simulation Variables (continued)

- Grid size: 60x60x1
- Grid dimensions: 20x10x5 ft
- Initial reservoir pressure: 3000 psia
- Initial reservoir saturation: $[0,1]$ - $[gas, water]$
- Total simulation time: 5 years
- Monitor steps: 1 month
- Total steps: 60
- Wellbore radius: 0.05
- AD Solver: TwoPhaseWaterGasModel

Proxy Model Variables

- L2 regularization: 1e-4
- LeakyReLU alpha: 0.3
- RNN dropout: 0.2
- Nadam learning rate: 5e-4

Proxy Model Code

```python
# Define proxy model by blocks
global_reg = 1e-4

# Convolutional block (Encoder)
def conv_block(filt, inp, kern=(3,3), reg=global_reg):
    x = Conv2D(filters=filt, kernel_size=kern, padding='same', activation=LeakyReLU(alpha=0.3))(inp)
    x = Conv2D(filters=filt, kernel_size=kern, padding='same', kernel_regularizer=regularizers.l2(reg))(x)
    x = BatchNormalization()(x)
    x = LeakyReLU(alpha=0.3)(x)
    x = MaxPooling2D(pool_size=(2,2))(x)
    return x

# Recurrent block
def rnn_block(units, inp, drop=0.2):
    x = Flatten()(inp)
    x = RepeatVector(n=125)(x)
    x = GRU(units=units, return_sequences=True, dropout=drop)(x)
    x = Reshape((5,5,5, x.shape[-1]))(x)
    return x

# Transpose Convolutional block (Decoder)
def convT_block(filt, stride, inp, kern=(3,3,3), reg=global_reg):
    x = Conv3DTranspose(filters=filt, kernel_size=kern, padding='same', strides=1,      activation=LeakyReLU(alpha=0.3))(inp)
    x = Conv3DTranspose(filters=filt, kernel_size=kern, padding='same', strides=stride, activation=LeakyReLU(alpha=0.3))(x)
    x = Conv3DTranspose(filters=filt, kernel_size=kern, padding='same', kernel_regularizer=regularizers.l2(reg))(x)
    x = BatchNormalization()(x)
    x = LeakyReLU(alpha=0.3)(x)
    return x

# Output block
def output_block(filt, inp, kern=(3,3,3)):
    x = Conv3DTranspose(filters=filt[0], kernel_size=kern, padding='same', activation='sigmoid')(inp)
    x = Conv3D(filters=filt[1], kernel_size=kern, padding='same', activation='sigmoid')(x)
    return x
```

```python
# Define CNN-RNN forward proxy model
def make_proxy():
    keras.backend.clear_session()

    # Input layer
    inp = Input(shape=(dim,dim,1))

    # Encoder block
    x = conv_block(filt=8,  inp=inp)
    x = conv_block(filt=16, inp=x)
    x = conv_block(filt=32, inp=x)
    x = conv_block(filt=64, inp=x)

    # Recurrent block
    x = rnn_block(units=128, inp=x)

    # Decoder block
    x = convT_block(filt=64, stride=2, inp=x)
    x = convT_block(filt=32, stride=2, inp=x)
    x = convT_block(filt=16, stride=3, inp=x)

    # Output block
    out = output_block(filt=[8,1], inp=x)

    proxy_model = Model(inp, out)
    return proxy_model
```