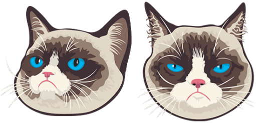
Speeding up wave propagation modeling
CheckPoint # 3

Team: GrumpyCat
Ivan Gerasimov, Olha Tsymboi

Moscow Institute of Physics and Technology

November 22, 2020



Introduction & Problem statement

Aim:

- To speed up acoustic equation solution using using neural networks.

Input:

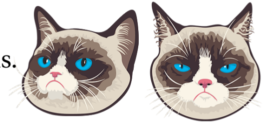
- time and space discretizations Δt, $\Delta x = \Delta z$
- time and space pints amout n_t, n_x, n_z
- impulse source time-series $q(t_i)$ and location x_s, z_s.
- special velocities $vp(x, z)$ at data points
- absorbing boundary conditions

Output:

- The solution of acoustic equation $u(x, z, t)$ at some point of time t.

Quality:

- RMSE error between normalized wavefields $\frac{u(x,z,t)}{\sigma(u)}$.
- Correlation coefficient between normalized wavefields.
- Execution time.



Data generation: velocity modeling

- Geo-realistic Marmousi-II model [1].
- Texture transferring using Random Gaussian field context [1].

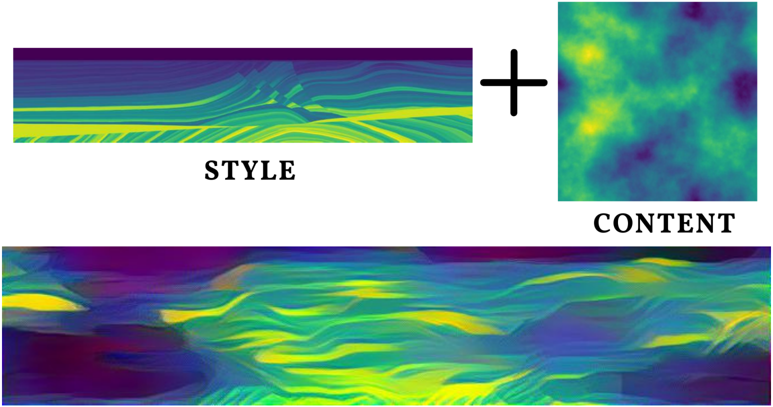


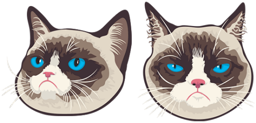
Figure: NST - style: Marmousi model, content: Gaussian field

Data generation: velocity modeling

- Geo-realistic scaling.

$$\left.\begin{aligned} &1)\ \mathrm{vp} = \mathrm{vp} - \mathrm{vp.min()} \\ &2)\ \mathrm{vp} = \frac{\mathrm{vp}}{\mathrm{vp.max()}} \end{aligned}\right\} \text{transform to scale } [0, 1]$$

$$\left.\begin{aligned} &[\mathrm{low},\ \mathrm{high}] \in [500, 10000] \\ &3)\ \mathrm{vp} = \mathrm{vp} * (\mathrm{high} - \mathrm{low}) + \mathrm{low} \end{aligned}\right\} \text{transform to scale } [\mathrm{low},\ \mathrm{high}]$$



Data generation

Generation:

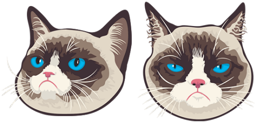
- Velocity models generation $vp(x,z)$:
- Nyquist frequency varying N_λ
- Space discretization varying $\Delta d,\ \Delta z = \Delta x = \Delta d$
- Ricker source with varying frequency f_0 under the condition

$$\begin{cases} f_0 & \in [6,\ 256] \\ \Delta x \cdot & \le \dfrac{\text{vp.min()}}{N_\lambda f_{\max}} \end{cases}$$

- Discretizations to fulfill CFL condition:

$$\Delta t \le \frac{\Delta x}{\sqrt{2}|vp(x,z)|.\max()}$$

- Random source locations



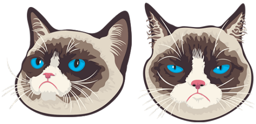
Model:

In the previous series

- Convolutional Auto-Encoder + L1Loss/MSE
- Convolutional Auto-Encoder + GRU Cell at the bottleneck + L1Loss/MSE
- UNet (UNet++)[8] + L1Loss/MSE
- Fully-Connected + Physic's Informed Loss [5]
- Convolutional + Physic's Informed Loss

Time is precious

- Avoid skip connections to reduce convolutions on large images
- Avoid point-wise prediction using FC layers
- Avoid extra encoder-decoder passes
  to predict full sequence



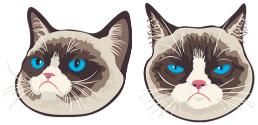
Model:

In the previous series

- Convolutional Auto-Encoder + L1Loss/MSE
- Convolutional Auto-Encoder + GRU Cell at the bottleneck + L1Loss/MSE
- UNet (UNet++)[8] + L1Loss/MSE
- Fully-Connected + Physic's Informed Loss [5]
- Convolutional + Physic's Informed Loss

Time is precious

- Avoid skip connections to reduce convolutions on large images
- Avoid point-wise prediction using FC layers
- Avoid extra encoder-decoder passes to predict full sequence



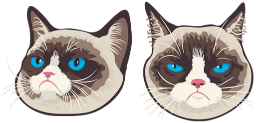
Model:

In the previous series

- Convolutional Auto-Encoder + L1Loss/MSE
- Convolutional Auto-Encoder + GRU Cell at the bottleneck + L1Loss/MSE
- UNet (UNet++)[8] + L1Loss/MSE
- Fully-Connected + Physic's Informed Loss [5]
- Convolutional + Physic's Informed Loss

Time is precious

- Avoid skip connections to reduce convolutions on large images
- Avoid point-wise prediction using FC layers
- Avoid extra encoder-decoder passes
  to predict full sequence



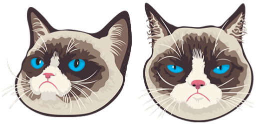
Model:

In the previous series

- Convolutional Auto-Encoder + L1Loss/MSE
- Convolutional Auto-Encoder + GRU Cell at the bottleneck + L1Loss/MSE
- UNet (UNet++)[8] + L1Loss/MSE
- Fully-Connected + Physic's Informed Loss [5]
- Convolutional + Physic's Informed Loss

Time is precious

- Avoid skip connections to reduce convolutions on large images
- Avoid point-wise prediction using FC layers
- Avoid extra encoder-decoder passes
  to predict full sequence



Model:

But still let us use Auto-Encoder as base-line

- Convolutional Auto-Encoder + L1Loss/MSE

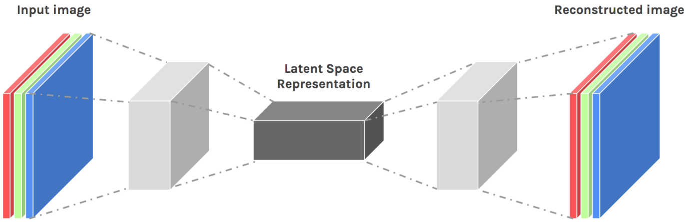
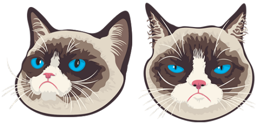


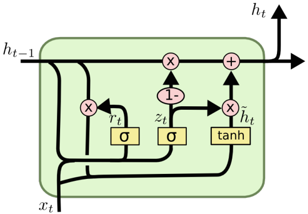
Figure: Convolutional Auto-Encoder

Input image: $u(x,z,t_i) + \frac{q(t_i)\Delta x^2}{\Delta t^2} \cdot \delta(x - x_s, z - z_s),\ vp(x,z)$



Model:

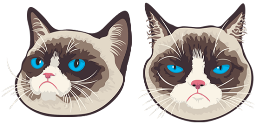
- Convolutional Auto-Encoder + GRU Cell at the bottleneck + L1Loss/MSE



$$z_t = \sigma(W_{hz} * h_{t-1} + W_{xz} * x_t + b_z)$$
$$r_t = \sigma(W_{hr} * h_{t-1} + W_{xr} * x_t + b_r)$$
$$\hat{h}_t = \Phi(W_h * (r_t \odot h_{t-1}) + W_x * x_t + b)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z \odot \hat{h}_t$$

Figure: Convolutional GRU Cell [7]

Input image: $u(x, z, t_i) + \frac{q(t_i)\Delta x^2}{\Delta t^2} \cdot \delta(x - x_s, z - z_s)$
Hidden initial image: $vp(x, z)$



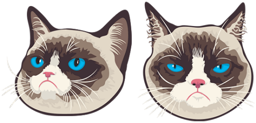
Model: Can we do faster?

In the sequential convolutions some weights could be too small to provide some influence on predictions

Thought-full weights thresholding

$$\widehat{W} = \text{ReLU}(W - f(s))$$

where s is a new trainable parameter and $f(\cdot)$ some nonlinearity [2]



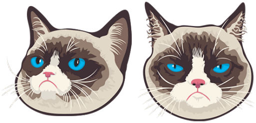
Model: Can we do faster?

In the sequential convolutions some weights could be too small to provide some influence on predictions

Thought-full weights thresholding

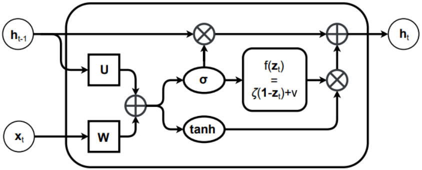
$$\widehat{W} = \mathrm{ReLU}(W - f(s))$$

where s is a new trainable parameter and $f(\cdot)$ some nonlinearity [2]



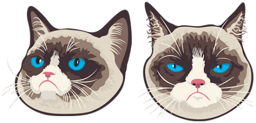
Model: Can we do faster?

Reduce GRU inner complexity



$$\mathbf{z}_t = \sigma(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}_z),$$
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b}_h),$$
$$\mathbf{h}_t = (\zeta(\mathbf{1} - \mathbf{z}_t) + \nu) \odot \tilde{\mathbf{h}}_t + \mathbf{z}_t \odot \mathbf{h}_{t-1},$$

Figure: FastGRNN Cell [3]



Results: CNN-AE + L1Loss

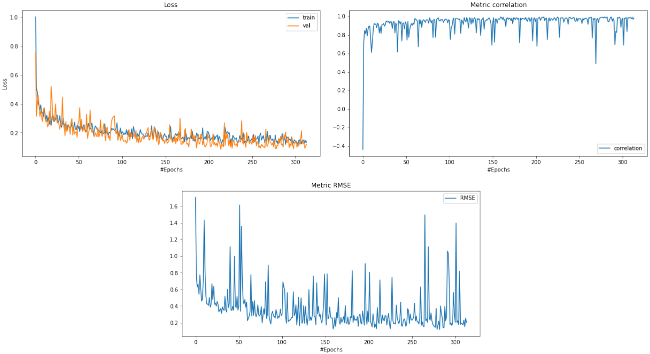
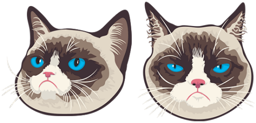
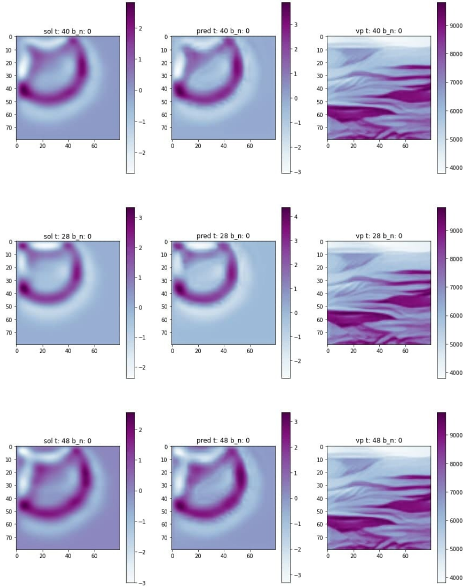
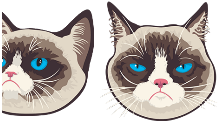


Figure: Tanh activations, MaxPool2d



Results: CNN-AE + L1Loss





Results: RNN-GRU + L1Loss

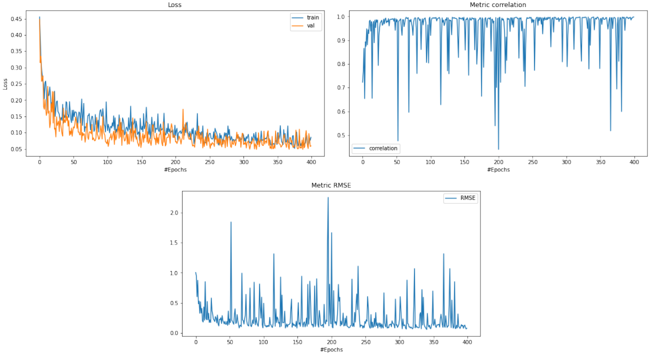
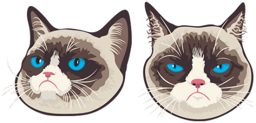
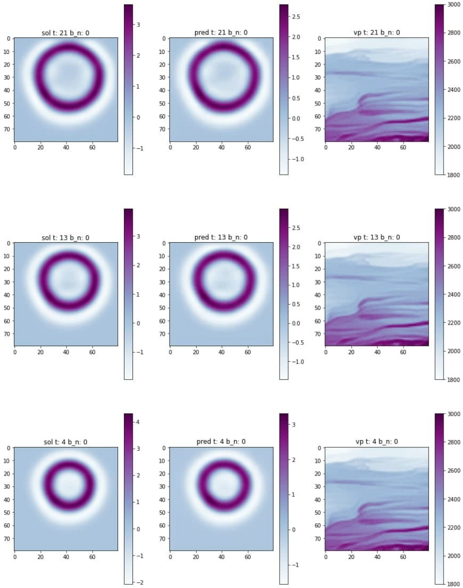
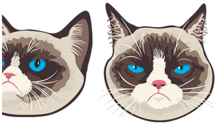


Figure: ELU activations, MaxPool2d




sol t: 21 b_n: 0
pred t: 21 b_n: 0
vp t: 21 b_n: 0
sol t: 13 b_n: 0
pred t: 13 b_n: 0
vp t: 13 b_n: 0
sol t: 4 b_n: 0
pred t: 4 b_n: 0
vp t: 4 b_n: 0
0 10 20 30 40 50 60 70
0 20 40 60
3000 2800 2600 2400 2200 2000 1800



Results: RNN-GRU + L1Loss

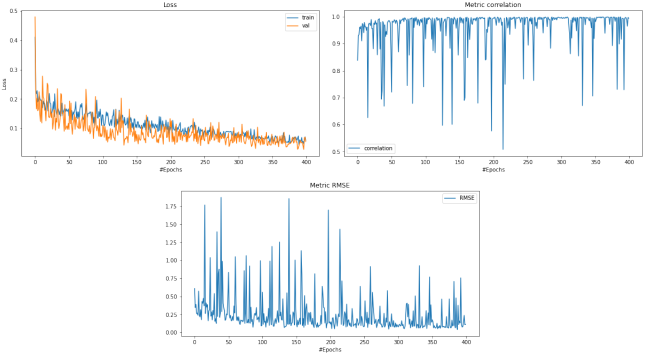
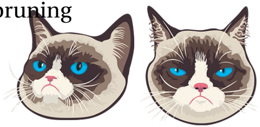
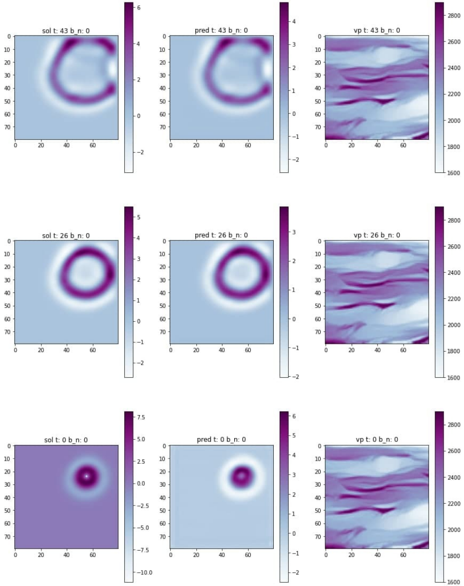
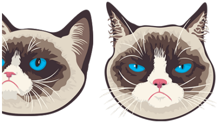


Figure: SoftPlus activations, AvgPool2d, Fast, pruning



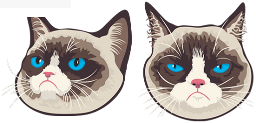
Results: CNN-AE + L1Loss





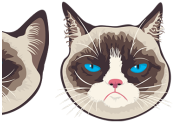
Results

	Val Loss	Correlation	RMSE
/AE/pruning_MaxPool2d_ELU_	0.047378	0.997431	0.071163
/AE/no_pruning_MaxPool2d_ELU_	0.040036	0.997193	0.078025
/AE/pruning_MaxPool2d_ReLU_	0.057838	0.996861	0.083887
/AE/pruning_AvgPool2d_ReLU_	0.051993	0.996960	0.083823
/AE/no_pruning_AvgPool2d	0.044700	0.996708	0.081910
/AE/no_pruning_AvgPool2d_Softplus_	0.055266	0.998026	0.076379
/AE/no_pruning_MaxPool2d_Soft	0.073278	0.997290	0.075836
/AE/no_pruning_MaxPool2d_ReLU_	0.067359	0.990594	0.132688
/AE/no_pruning_AvgPool2d_ReLU_	0.054532	0.997037	0.081428
/AE/no_pruning_AvgPool2d_Tanh_	0.108479	0.993235	0.116443
/AE/no_pruning_MaxPool2d_Tanh_	0.084698	0.992608	0.117035



Results

	Val Loss	Correlation	RMSE
/RNN/pruning_MaxPool2d_ELU_	0.054459	0.996234	0.084615
/RNN/pruning_AvgPool2d_ELU_	0.061728	0.997185	0.072924
/RNN/pruning_AvgPool2d_ELU_fooo__	0.056141	0.997540	0.068849
/RNN/pruning_AvgPool2d_Softplus_fast__	0.028346	0.999053	0.041984
/RNN/no_pruning_MaxPool2d_ELU_fooo__	0.048708	0.998025	0.056068
/RNN/pruning_MaxPool2d_ReLU_fooo__	0.044532	0.997466	0.068267
/RNN/pruning_AvgPool2d_ReLU_fast_	0.040655	0.998074	0.057065
/RNN/no_pruning_AvgPool2d_ELU_fooo__	0.029321	0.999008	0.043311
/RNN/pruning_MaxPool2d_Tanh_fast_	0.101255	0.991704	0.141311
/RNN/no_pruning_AvgPool2d_Tanh_fast__	0.058024	0.995073	0.082453
/RNN/pruning_MaxPool2d_ReLU_fast_	0.064585	0.995742	0.091423
/RNN/no_pruning_AvgPool2d_Tanh_fooo__	0.051936	0.998001	0.057703
/RNN/no_pruning_MaxPool2d_Tanh_fast_	0.097993	0.989757	0.120297
/RNN/no_pruning_AvgPool2d_ELU_fast__	0.075362	0.993769	0.107612
/RNN/no_pruning_MaxPool2d_Softplus_fast_	0.055147	0.996729	0.085772
/RNN/no_pruning_AvgPool2d_Softplus_fast__	0.072308	0.992906	0.116746
/RNN/no_pruning_MaxPool2d_ELU_fast_	0.094881	0.991725	0.136576
/RNN/no_pruning_AvgPool2d_ReLU_fast_	0.084444	0.986034	0.149476
/RNN/no_pruning_MaxPool2d_Tanh_fooo__	0.064734	0.995489	0.093585
/RNN/pruning_AvgPool2d_Softplus_fooo__	0.056134	0.998492	0.059784
/RNN/no_pruning_MaxPool2d_Softplus_fooo__	0.056357	0.997816	0.064815



Further work

- Additional restriction on hiddens in GRU model

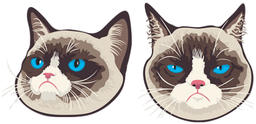
$$L_{total} = L(u_{t+1}, NN(u_t)) + \lambda L_{hidden}(h_{t+1}(u_t), encoder(u_{t+1}))$$

- FC pruning for Physic's Informed Loss net

$$W = U\Sigma V^\top$$

$$\hat{\Sigma} = \mathrm{ReLU}(\Sigma - f(s))$$

$$\widehat{W} = U\hat{\Sigma}V^\top$$



References I

D. Bevc and O. Nedorub.
SEG Technical Program Expanded Abstracts 2019.
Society of Exploration Geophysicists, 2019.

A. Kusupati, V. Ramanujan, R. Somani, M. Wortsman, P. Jain, S. Kakade, and A. Farhadi.
Soft threshold weight reparameterization for learnable sparsity, 2020.

A. Kusupati, M. Singh, K. Bhatia, A. Kumar, P. Jain, and M. Varma.
Fastgrnn: A fast, accurate, stable and tiny kilobyte sized gated recurrent neural network, 2019.

M. Louboutin, M. Lange, F. Luporini, N. Kukreja, P. A. Witte, F. J. Herrmann, P. Velesko, and G. J. Gorman.
Devito (v3.1.0): an embedded domain-specific language for finite differences and geophysical exploration.
Geoscientific Model Development, 12(3):1165–1187, 2019.

B. Moseley, A. Markham, and T. Nissen-Meyer.
Solving the wave equation with physics-informed deep learning, 06 2020.

References II

A. Siahkoohi, M. Louboutin, and F. Herrmann.
Neural network augmented wave-equation simulation, 09 2019.

M. Siam, S. Valipour, M. Jagersand, and N. Ray.
Convolutional gated recurrent networks for video segmentation, 2016.

Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang.
Unet++: Redesigning skip connections to exploit multiscale features in image segmentation, 2020.

Thank you!
Questions?