Wavefield compression for seismic imaging via convolutional neural networks

Francesco Devoti, Claudia Parera, Alessandro Lieto, Daniele Moro, Vincenzo Lipari, Paolo Bestagini and Stefano Tubaro, Politecnico di Milano, Italy

SUMMARY

Full Waveform Inversion and Reverse Time Migration are usually based on the adjoint state method and rely on the ready availability of wavefield snapshots. Therefore, standard algorithms store the full source wavefield on disk. This makes Full Waveform Inversion and Reverse Time Migration techniques particularly demanding in terms of disk input/output performance. To face this issue, a common solution is to compress wavefield information in order to reduce input/output operations overhead.
In this paper we propose a couple of wavefield compression methods based on Convolutional Neural Networks (CNNs). Specifically, a convolutional autoencoder is trained to compress wavefield snapshots and a specifically designed *U-net* is trained to reconstruct (i.e. interpolate) the wavefield in the temporal dimension.
Results show that these promising techniques could help decreasing storage needs for wavefield snapshots. This makes it possible to balance the required signal-to-noise ratio and compression gain.

INTRODUCTION

State of the art imaging technologies, such as Full Waveform Inversion (FWI) (Plessix, 2006) and Reverse Time Migration (RTM) often rely on the adjoint state method. In these imaging procedures both forward and adjoint wavefields are needed, at each time step, for the computation of sensitivity kernels, and this is usually obtained by saving the whole forward wavefield thus requiring huge storage capabilities. Moreover, this introduces an important bottleneck because storing the whole forward wavefield to the disk causes significant I/O overhead and dramatically worsens the performances, especially on GPUs.

The main strategies proposed to attenuate this problem are based on checkpointing and compression. Checkpointing techniques, that reduce the memory requirements but greatly increase the computational cost, have been successfully applied in RTM and FWI (Symes, 2007; Anderson et al., 2012). The typical overhead caused by this approach is the order of one additional forward simulation.

A valid alternative for reducing storage requirements and I/O overhead is given by compression techniques. In an ideal case, if we were able to compress the forward wavefield enough to fit it entirely into the memory, neither disk I/O nor checkpointing would be necessary anymore.

Industrial solutions often use standard lossless data compression algorithms. Lossless compression allows an exact data reconstruction, whereas lossy algorithms introduce a controlled error, but are more effective in terms of compression ratio (CR). The straightforward solution to recast data into a lower precision format (i.e. from double to single precision), which is often used in industrial solutions, is nothing but a trivial lossy compression scheme. Other lossy compression algorithms, tailored for scientific floating-point data, have been recently developed (Di et al., 2018). Dalmau et al. (2014) and Boehm et al. (2016) proposed compression algorithms specifically designed for the wavefields simulated in FWI.

The main innovation of the last few years, in the signal and data processing field, is certainly the explosion of techniques based on CNNs and deep learning, that recently had a great impact also in the seismic data processing and imaging community. Interesting results have been obtained for seismic processing tasks such as interpolation (Mandelli et al., 2018), denoising (Li et al., 2018), processing of seismic images (Picetti et al., 2018) and velocity model building (Wang et al., 2018). In particular, the important results achieved for denoising and interpolation suggest that CNNs have the capability to provide a compact representation of seismic data.

These considerations provide the reasons to explore CNNs as a tool for lossy compression of seismic wavefields. In particular, in this work we study two CNN based compression strategies for seismic wavefields, namely snapshots compression and snapshots interpolation. These strategies can be also jointly combined to improve compression results. To perform snapshots compression we leverage a specially designed convolutional autoencoder that projects data into a reduced dimensionality space. The strategy based on interpolation consists in selecting a series of reference snapshots by subsampling the wavefield in the temporal dimension. We leverage a *U-net* CNN to interpolate reference snapshots in time and estimate the decimated ones.

The promising results obtained on 2D propagation suggest to further explore CNNs as a viable alternative for seismic wavefield compression in FWI and RTM.

WAVEFIELD COMPRESSION VIA CONVOLUTIONAL NEURAL NETWORKS

In adjoint state methods we first perform forward propagation and store the propagating wavefield. The resulting synthetic data is combined with the acquired data to build an adjoint source which is then back-propagated. At each time step, the back-propagating snapshot is cross-correlated with the corresponding saved snapshot of the forward wavefield. Ideally, all these operations should not require access to mass storage. Therefore, both the compressed forward wavefield and the decompressed snapshot should fit into the memory. This means that a compression scheme suitable for this task should ideally allow streaming decompression or at least partial decompression.

For this reason we explored compression strategies pricipally based on snapshots. In particular, let $u(x,y,t)$ be the wavefield to compress, where t indicates the temporal dimension, while x and y are spatial coordinates. For the sake of simplicity, we can describe the wavefield as a set of snapshots $\{\mathbf{u}_t\}$, with $t \in \{0,\dots,N\}$.

The strategies studied in this paper are snapshot compression via convolutional autoencoder and snapshot decimation and interpolation via CNN.

Snapshots compression via convolutional autoencoders

Autoencoders are neural networks that attempt to copy their inputs to their outputs through learning (Goodfellow et al., 2016). They can be logically split in the following components:

(i) Encoder: Is a convolutional neural network which maps the input $\mathbf{u}_t$ into the hidden representation, $\theta_t = f(\mathbf{u}_t)$. This is the component that performs data compression if the dimensionality of θ_t is smaller than $\mathbf{u}_t$.

(ii) Bottleneck: Is the central layer of the autoencoder which contains the hidden representation, i.e. the encoded version of the input, θ_t.

(iii) Decoder: Is a convolutional neural network which reconstructs the input from the hidden representation, i.e. $\hat{\mathbf{u}}_t = g(\theta_t)$. This component performs the decoding.

In convolutional autoencoders, both the encoder and decoder operators are composed by series of linear filtering operations (i.e., convolutions), generally followed by non-linear activation functions (e.g., sigmoid, hyperbolic tangent, etc.) and, optionally, other linear and non-linear layers.

In order to use an autoencoder for snapshots compression we divided the snapshot in patches, thus allowing us to perform compression of snapshots of any dimension and to enlarge the training dataset. In particular, after a preliminary experimental campaign we set a patch size of 128×128 patches for the autoencoder studied in this paper. The first and last layers are comprised of a 2D convolution with filter size 6×6 and stride 1×1. The remaining encoding and decoding layers are comprised of four 2D convolution and four 2D deconvolution layers with decreasing filter size and stride 2×2.

To train the autoencoder, we use as loss function to be minimized a weighted average between the Mean Squared Error (MSE) (forcing similarity between input and output) and the *L1* norm of the encoded snapshot (increasing the sparsity of the hidden representation). Formally:

$$\mathscr{L}(\mathbf{u}_t, \hat{\mathbf{u}}_t, \theta_t) = E[(\mathbf{u}_t - \hat{\mathbf{u}}_t)^2] + \alpha||\theta_t||_1. \quad (1)$$

By using this architecture we achieve a compression ratio of 32 just considering dimensionality reduction from input to hidden representation.

Moreover, we introduce a threshold Γ to cut-off small hidden representation values. This allows us to further increase the compression ratio via the lossless Lempel-Ziv-Markov (Zip) algorithm.

Snapshots interpolation via U-net

U-net (Ronneberger et al., 2015) is a special autoencoder architecture that adds *shortcuts* between *encoder* and *decoder* paths. Specifically, each layer of the encoder is also connected directly, trough the so-called skip-connections, with the corresponding layer on the decoder. This architecture, originally designed for the image segmentation task, shows good performance also for image denoising and inpainting.

Here again we divide the input snapshots in patches, of size 128×128. In particular, we feed the network with patches extracted, at the same location, from a couple of triplets of snapshots, $\{\mathbf{u}_{n\mathrm{k}}, \mathbf{u}_{n\mathrm{k}+1}, \mathbf{u}_{n\mathrm{k}+2}\}$ and $\{\mathbf{u}_{(n+1)\mathrm{k}}, \mathbf{u}_{(n+1)\mathrm{k}+1}\mathbf{u}_{(n+1)\mathrm{k}+2}\}$. Each patch is normalized between 0 and 1 before being fed into the network.

The input layer takes the two triplets of patches. The encoding path is comprised of seven stages where a 2D convolution with filter size 4×4 and stride 2×2, followed by batch normalization and Leaky ReLu, are performed. As we go deep the number of filters increases from 64 to 512, obtaining a 512 elements *bottleneck*. The decoding path is comprised of seven stages where a ReLu, an up-sampling with factor 2, and a 2D convolution with filter size 4×4 and stride 1×1 followed by batch normalization are performed. In each stage we concatenate the result coming from the corresponding encoding stage. As we go up in the decoding path of the network, the number of filters is gradually diminished from 512 to 64. The output layer is comprised of $k-3$ reconstructed patches.

The network is trained exploiting the original $k-3$ snapshots for estimating the *U-net* parameters. The model weights are estimated by minimizing a loss function corresponding to the MSE between reconstructed and original snapshots. The rationale behind the actual input choice is that the ideal interpolation should simulate the behaviour of the wave equation, which is driven by spatial and temporal 2^{nd} derivatives. Moreover, feeding the *U-net* with triplets of snapshots will implicitly allow the CNN to learn from 2^{nd} order time derivatives.

Joint compression/interpolation

The aforementioned CNN based procedures outline a potential workflow allowing to jointly exploit both techniques for wavefield compression.

First, the wavefield is subsampled in the temporal dimensions in order to select bursts of three adjacent snapshots every k ($\{\mathbf{u}_{n\mathrm{k}}, \mathbf{u}_{n\mathrm{k}+1}, \mathbf{u}_{n\mathrm{k}+2}\}$, $\{\mathbf{u}_{(n+1)\mathrm{k}}, \mathbf{u}_{(n+1)\mathrm{k}+1}, \mathbf{u}_{(n+1)\mathrm{k}+2}\}\dots$, with $n \in \{0,\dots,\frac{N-2}{k}\}$) and to discard the others. Then, the snapshots are compressed through the autoencoder specially trained for this task. At decoding time, compressed snapshots are decoded, and fed into a specifically designed *U-net*. The *U-net* is used to estimate the remaining snapshots (the $k-3$ discarded during compression), through interpolation.

The resulting compression/decompression procedure could be summarized by the following pipeline:

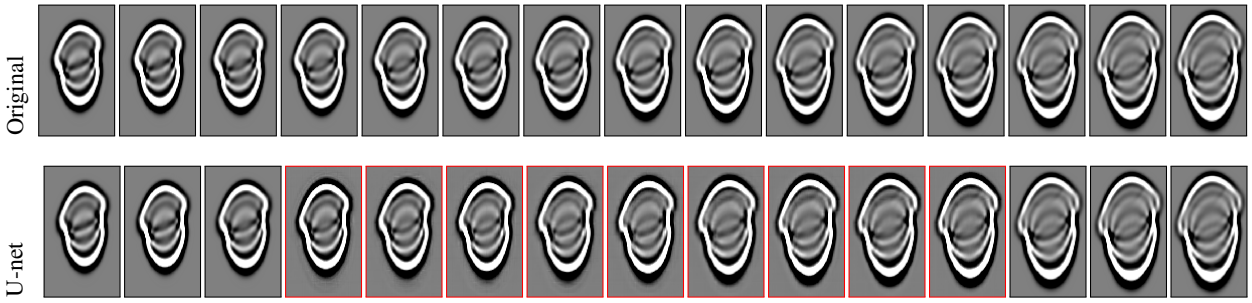


Figure 1: Example of original snapshots compared to snapshots interpolated trough the proposed *U-net*

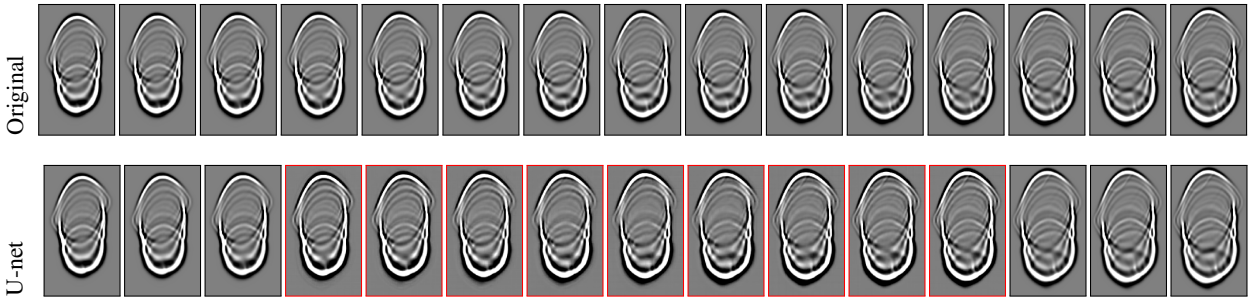


Figure 2: Example of original snapshots compared to snapshots interpolated trough the proposed *U-net*

Compression:

i The wavefield is decimated keeping only one triplet $\{\mathbf{u}_{n\mathrm{k}}, \mathbf{u}_{n\mathrm{k}+1}, \mathbf{u}_{n\mathrm{k}+2}\}$ of neighboring snapshots every k.
ii The resulting triplet $\{\mathbf{u}_{n\mathrm{k}}, \mathbf{u}_{n\mathrm{k}+1}, \mathbf{u}_{n\mathrm{k}+2}\}$ is compressed through the encoding branch of the autoencoder obtaining $\theta_{n\mathrm{k}}, \theta_{n\mathrm{k}+1}, \theta_{n\mathrm{k}+2}$.
iii Thresholding and lossless compression are then applied.

Decompression:

i Lossless decompression is applied to output the encoded snapshots.
ii The encoded snapshots $(\theta_{n\mathrm{k}}, \theta_{n\mathrm{k}+1}, \theta_{n\mathrm{k}+2})$ are decompressed through the decoding branch of the autoencoder, obtaining the corresponding decompressed snapshots $\{\hat{\mathbf{u}}_{n\mathrm{k}}, \hat{\mathbf{u}}_{n\mathrm{k}+1}, \hat{\mathbf{u}}_{n\mathrm{k}+2}\}$.
iii By feeding the trained *U-net* with a couple of adjacent triplets of decompressed snapshots $\{\hat{\mathbf{u}}_{n\mathrm{k}}, \hat{\mathbf{u}}_{n\mathrm{k}+1}, \hat{\mathbf{u}}_{n\mathrm{k}+2}\}$ and $\{\hat{\mathbf{u}}_{(n+1)\mathrm{k}}, \hat{\mathbf{u}}_{(n+1)\mathrm{k}+1}, \hat{\mathbf{u}}_{(n+1)\mathrm{k}+2}\})$, the $k-3$ snapshots between them are reconstructed.

The rationale behind the outlined strategy is that a properly designed autoencoder allows effective compression of the snapshots by projecting them into a reduced dimensionality space, whereas the *U-net* guarantees an additional $\mathrm{k}/3$ compression ratio at the cost of a minimum decompressed atom of $k+3$ snapshots.

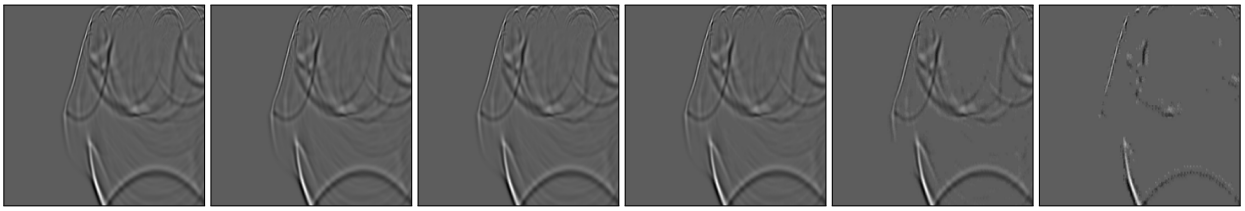
EXPERIMENTAL RESULTS

In this initial study we investigated the two proposed compression procedures separately. In order to build the dataset used to test both strategies we proceeded as follows:

i We extracted 7 velocity models of 1024×1024 samples extracted from the well-known BP-2004 and Marmousi models with spatial sampling rates $\Delta x = 10$m and $\Delta z = 5$m). The models were selected to mimic different scenarios (smooth models, layering, faults, salt bodies, etc.).
ii For each model we simulated 5 equispaced sources at depth $z = 0$m and 5 equispaced sources at depth $z = 2560$m (at the center of the velocity model in depth).
iii For each source we computed 4 seconds of propagation through a finite differences code based on Devito (Luporini et al., 2018) and we stored a snapshot every $dt = 0.01$s.

In this way we produced a dataset made of 70 volumes of propagating wavefield, with 400 snapshots each, for a total of 28000 snapshots.

To train the *U-net* for wavefield interpolation we selected 15 volumes of propagating wavefield which we further split in a 75% testing dataset and a 25% validation dataset. The training was performed through 50 epochs of the Adam optimization algorithm on batches of 64 patches. The learning rate is initialized at 0.01 and it is subsequently halved in the presence of a plateau of the cost function. In Figures 1 and 2 we show



(a) Original (b) $SNR = 25.30$ dB, $CR = 113.52$ (c) $SNR = 25.24$ dB, $CR = 138.19$ (d) $SNR = 24.29$ dB, $CR = 189.28$ (e) $SNR = 21.16$ dB, $CR = 348.30$ (f) $SNR = 14.07$ dB, $CR = 825.81$

Figure 3: Example of original snapshots vs snapshots reconstructed by the proposed autoencoder.

two examples of snapshot interpolation through the proposed *U-net* with k set to 12. In both figures the snapshots at the top display the ground truth (i.e. the actual propagating wavefield), while the ones at the bottom display the corresponding *U-net* outputs. In particular, the snapshots highlighted in red represent the $k-3$ interpolated snapshots. Figure 1 show how the interpolated snapshots are able to reconstruct an expanding wavefield. Figure 2 shows the interpolation results on a quite complex wavefield that would be difficult to reconstruct by a standard interpolation strategy.

The overall Signal-to-Noise Ratio (SNR) achieved on the testing dataset for the interpolated snapshots via *U-net* is always above 40 dB.

As for the autoencoder based compression strategy, the training was performed on a set 10000 snapshots (75% training set and 25% validation set), with 50 epochs of Adam optimization and a batch size of 32.

The performances of the network in terms of Compression Ratio (CR) and SNR are driven by the parameters α (i.e. the regularization weight for the loss function enhancing sparsity of the hidden representation) and Γ (i.e. the threshold for the hidden representation values). Given a desired SNR, we heuristically selected the best configuration of α and Γ that maximizes the CR, obtaining the average results shown in Figure 4. We observed a linear link between the increase of CR and the decrease in SNR.

In Figure 3 we show an example of autoencoder reconstruction results for different values of SNR and achieved compression ratio (CR), which is a visual proof of the aforementioned behavior. Notice that compression ratios superior to 100 still guarantee an SNR superior to 25 dB.

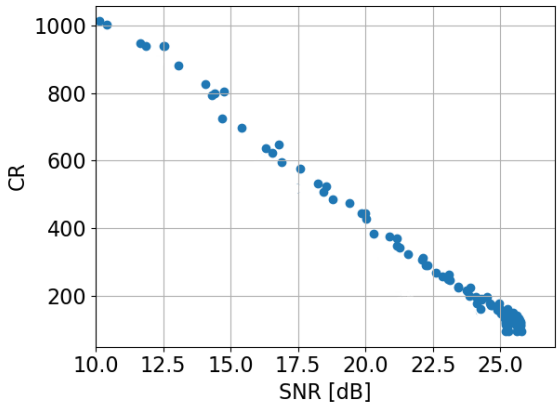


Figure 4: SNR vs Compression Ratio in auotencode based compression

CONCLUSIONS

In this extended abstract, we describe a preliminary study for a CNN-based pipeline for wavefield compression. Specifically, we studied two different procedures: a convolutional autoencoder used to separately compress snapshots, and another CNN architecture (i.e. a *U-net*) that interpolate wavefields in time. These two methods can be combined together to further increase the compression performances.

Although this is a preliminary work, the promising results obtained on 2D propagation suggest that CNNs could represent a viable alternative for wavefield compression and pave the way to further investigations.

Future studies will be devoted to design more specific CNN architectures for compression and interpolation of wavefields and to extend the method to 3D propagation, which will allow us to use a more sparse representation and larger compression ratios. Moreover, more sophisticated quantization strategies and lossless compression schemes for the encoded representation could greatly improve the compression performances. Future work will also include the integration of CNN based compression techniques into FWI and RTM algorithms.

ACKNOWLEDGEMENTS

The authors would like to thank Nicola Bienati for the inspiring discussions.

REFERENCES

Anderson, J. E., L. Tan, and D. Wang, 2012, Time-reversal checkpointing methods for rtm and fwi: Geophysics, **77**, S93–S103.

Boehm, C., M. Hanzich, J. de la Puente, and A. Fichtner, 2016, Wavefield compression for adjoint methods in full-waveform inversion: Geophysics, **81**, R385–R397.

Dalmau, F. R., M. Hanzich, J. de la Puente, and N. Gutiérrez, 2014, Lossy data compression with dct transforms: Presented at the EAGE Workshop on High Performance Computing for Upstream.

Di, S., D. Tao, X. Liang, and F. Cappello, 2018, Efficient lossy compression for scientific data based on pointwise relative error bound: IEEE Transactions on Parallel and Distributed Systems.

Goodfellow, I., Y. Bengio, A. Courville, and Y. Bengio, 2016, Deep learning: MIT press Cambridge, **1**.

Li, H., W. Yang, and X. Yong, 2018, Deep learning for ground-roll noise attenuation, *in* SEG Technical Program Expanded Abstracts 2018: Society of Exploration Geophysicists, 1981–1985.

Luporini, F., M. Lange, M. Louboutin, N. Kukreja, J. Hückelheim, C. Yount, P. Witte, P. H. J. Kelly, G. J. Gorman, and F. J. Herrmann, 2018, Architecture and performance of devito, a system for automated stencil computation: CoRR, **abs/1807.03032**.

Mandelli, S., F. Borra, V. Lipari, P. Bestagini, A. Sarti, and S. Tubaro, 2018, Seismic data interpolation through convolutional autoencoder, *in* SEG Technical Program Expanded Abstracts 2018: Society of Exploration Geophysicists, 4101–4105.

Picetti, F., V. Lipari, P. Bestagini, and S. Tubaro, 2018, A generative adversarial network for seismic imaging applications, *in* SEG Technical Program Expanded Abstracts 2018: Society of Exploration Geophysicists.

Plessix, R.-E., 2006, A review of the adjoint-state method for computing the gradient of a functional with geophysical applications: Geophysical Journal International, **167**, 495–503.

Ronneberger, O., P. Fischer, and T. Brox, 2015, U-net: Convolutional networks for biomedical image segmentation: International Conference on Medical image computing and computer-assisted intervention, Springer, 234–241.

Symes, W. W., 2007, Reverse time migration with optimal checkpointing: Geophysics, **72**, SM213–SM221.

Wang, W., F. Yang, and J. Ma, 2018, Velocity model building with a modified fully convolutional network, *in* SEG Technical Program Expanded Abstracts 2018: Society of Exploration Geophysicists, 2086–2090.