Stepped Frequency Pulse Compression with Non-Coherent Radar using Deep Learning

Alexander Karlsson, Magnus Jansson, and Henrik Holter

***Abstract*—A deep neural network (DNN) is used for achieving subpulse resolution in non-coherent stepped frequency waveform radar. The trade-off between high resolution and long range in radar systems is often addressed using pulse compression, allowing both long pulses and high resolution by increasing the pulse bandwidth. This typically requires a coherent radar. In this study we present a deep learning based solution for achieving subpulse resolution with a non-coherent radar. Our results for such a system are comparable to an equivalent coherent system for SNRs greater than 10 dB. All results are based on simulated data.**

***Index Terms*—Deep learning, supervised learning, non-coherent radar, pulse compression, frequency-agile radar**

I. Introduction

A key design parameter of any radar system is its resolution which has a theoretical limit [1]

$$\sigma_r = \frac{c}{2B}, \tag{1}$$

where c is the speed of light and B is the bandwidth of the transmitted pulse. For a single frequency rectangular pulsed radar, the -4 dB bandwidth is determined by the inverse of the pulse length, τ. The bandwidth can be increased by various pulse modulation techniques, known as pulse compression. It aims to increase both range resolution and signal-to-noise ratio (SNR). As this often requires phase information the term "pulse compression" is almost synonymous with coherent radar.

There are however methods for non-coherent pulse compression, first introduced in [2] and further developed in [3], [4], [5], where the pulse is divided into a series of sub-pulses acting as on/off bits, referred to as on-off keying (OOK). When cross-correlated with a reference sequence a single peak/pulse is obtained with increased SNR. The resolution is determined by the length of each sub-pulse and the gain in SNR depends on the number of sub-pulses, i.e. having only one sub pulse is equivalent to a standard pulsed radar. According to [6] such non-coherent pulse compression performs slightly worse in terms of SNR gain than simply integrating several consecutive pulses but with the advantage of getting the gain in one pulse repetition interval (PRI).

In this study we present an alternative method to non-coherent pulse compression with the main purpose of increasing the range resolution without increasing the instantaneous bandwidth, i.e. maintaining a relatively long, uninterrupted pulse length. In other words, we consider radar systems with constraints on instantaneous bandwidth and with no phase information. As for constraints on instantaneous bandwidth, a common waveform for coherent systems is the stepped frequency waveform. This utilizes frequency agility where pulses are transmitted at different frequencies, yielding an increased total bandwidth over a set of pulses, and thereby higher range resolution (1). In this study we use a non-coherent stepped frequency waveform and a deep neural network (DNN) acting as a pulse compression "filter."

Unlike SNR gain, a resolution gain can be of significant value even at high SNRs. As an example, an SNR of 18 dB yields a detection probability of 99.99% and false alarm probability of 10^{-10} for a Rayleigh distributed target in additive white Gaussian noise (AWGN) [7, p. 17]. Increasing the SNR beyond that will thus only yield marginal improvements in the detection probability and false alarm trade-off. The resolution gain is however valuable regardless of SNR. Note that for coherent systems the term "pulse compression" implies both these gains.

The method we present may be particularly suitable for non-coherent radar systems that require frequency agility for other reasons, such as minimizing interference, as it in such cases may only require additional signal processing rather than additional physical requirements on the radar system. Obviously non-coherent systems will always be inferior to their coherent counterparts in terms of performance. The reason for choosing a non-coherent system in the first place is instead based on other factors such as production cost, power criteria, and design complexity.

With the stepped frequency waveform and a coherent radar system, a high resolution range profile is achieved by taking the inverse discrete Fourier transform (IDFT) of a set of M pulses [8]. See Section II-B. In a non-coherent case, we only have the magnitude of each pulse sample, resulting in an inverse phase problem where we aim to transform a complex vector from frequency to time/range domain without knowing the phase, $|\mathbf{G}(f)|^2 \rightarrow \mathbf{g}(t)$. As is the case here, this problem often arises with non-coherent radiation, i.e. unknown phase, such as X-ray and optics and has been widely studied; see e.g.

Submitted for review on April 28 2020. This work was supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP), SAAB Electronic Warfare Systems, the Division of information science and engineering at KTH, and the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 742648).

Alexander Karlsson is with the Division of Information Science and Engineering at KTH Royal Institute of Technology and SAAB Electronic Warfare Systems, Stockholm Sweden (e-mail: alek@kth.se)

Magnus Jansson is with the Division of Information Science and Engineering at KTH Royal Institute of Technology, Stockholm Sweden (e-mail: janssonm@kth.se)

Henrik Holter is with SAAB Electronic Warfare Systems, Stockholm Sweden (e-mail: henrik.holter@saabgroup.com)

[9]. Iterative algorithms exist for going from $|\mathbf{G}(f)|^2$ to $\mathbf{g}(t)$, [10], and require some prior knowledge of $\mathbf{g}(t)$, e.g. that it is non-negative and real, or that the signal is sparse. Without prior knowledge of the time domain signal the problem is impossible to solve since any random guess of the phase is then a valid solution. In our case the target is complex-valued, see Section II-B, making the iterative solutions difficult to apply. Moreover, such iterative methods may be sensitive to initial values, and require an undetermined number of iterations.

Non-iterative and non-DNN based solutions for sparse phase retrieval have also been developed where K scattering points are resolved from the magnitude of the frequency spectrum [11]. However, the required number of samples in frequency domain is on the order K^2, which in this context may be very large.

Since we are mainly interested in retrieving the magnitude of $\mathbf{g}(t)$, we will instead set up a non-linear regression problem where we train a neural network to directly transform a frequency domain matrix $\mathbf{X}$ to a discrete estimate of the magnitude in the time domain, $\mathbf{y}$. We thereby never explicitly retrieve the phase. Since the execution time of a trained neural network is deterministic, this solution may be preferable over the aforementioned iterative solutions, an aspect that has been further studied in [12].

Although the use of deep learning has been studied in the field of phase retrieval, e.g. [12], [13], [14], and [15], it has to our knowledge not been applied in this context of non-coherent radar, and unlike most other such studies we use a DNN as a direct magnitude-to-magnitude transformation, and do not recover the phase. These two solutions are only equivalent when the phase reconstruction is perfect.

A drawback with neural networks is that they require large amounts of training data to generalize well. To manage this, we generate data from a model to train the network on. Our results are therefore constrained to the validity of this model. By using data from a model we have complete control of the data and associated labels, or ground truth, which is not necessarily the case when training on real data. The rest of this paper is organized as follows. In Section II the model for generating $\mathbf{X}$ and $\mathbf{y}$ as well as the network structure is presented. Simulation results are shown in Section III and conclusions and directions for future research are given in Section IV.

II. System Model

A. Signal model

We model all targets as a collection of point scatterers. Furthermore we will assume a two dimensional stationary scenario with the radar as the reference point. The received signal from all K scatterers at range r_i and at frequency f_c+f_m can then, after complex down conversion and lowpass filtering, be modelled as [16, p. 529]

$$v(f_m, r_i) \propto \lambda_m e^{j\phi_m} \sum_{k=0}^{K-1} s_k g_{k,i} e^{-j2\pi(f_c+f_m)\frac{2\delta_k}{c}} + w_m, \quad (2)$$

where

- $v(f_m, r_i) = v_{m,i}$ has unit Volt
- $\lambda_m = c/(f_c + f_m)$, and accounts for the antenna gain dependence on frequency.
- ϕ_m is a random phase for pulse m.
- f_c is the carrier frequency
- f_m is a deviation from the carrier frecuency in pulse m and is confined to a bandwidth B, i.e. $f_m \in [-B/2, B/2]$
- c is the speed of light
- δ_k is the radial range to scatterer k
- s_k is the amplitude of the k'th scatterer and is proportional to the square root of its Radar Cross Section (RCS) and $1/\delta_k^2$
- $g_{k,i}$ is the pulse and beamwidth gain at scatterer k and is a function of the difference between the radar line of sight angle φ, and the angle to the scatterer, α_k, as well as the difference between δ_k and the sampled range r_i (or in time t_k and t_i). The range r_i is defined as the range to the beginning of range cell i.
- w_m is complex zero mean Gaussian white noise with variance σ_w^2 in each dimension.

Let our stepped frequency waveform be such that

$$f_m = \frac{N-1}{\tau'}\left(\frac{m}{M-1} - \frac{1}{2}\right) \quad (3)$$

for pulses $m = 0, ..., M-1$, where N determines the total bandwidth $B \approx N/\tau'$, $M \geq 2$, $M \geq N$, and τ' is the full pulse duration of a bandwidth limited pulse, not the 3 dB length. Inserting (3) in (2) and multiplying by $1/\lambda_c$ gives

$$v_{m,i} \propto \beta_m e^{j\phi_m} \sum_{k=0}^{K-1} s_k g_{k,i} e^{-j2\pi\alpha_m \frac{\delta_k}{\Delta r}} + w_m \quad (4)$$

where

$$\beta_m = \frac{1}{1+f_m/f_c} = \frac{1}{1+\frac{\lambda_c}{2\Delta r}\left(m-\frac{M-1}{2}\right)\frac{N-1}{M-1}} \quad (5)$$

$$\alpha_m = \frac{2\Delta r}{\lambda_c} + (N-1)\left(\frac{m}{M-1} - \frac{1}{2}\right) \quad (6)$$

$$\Delta r = \frac{c\tau'}{2}. \quad (7)$$

This formulation allows us to generate $v_{m,i}$ with all distances normalized to the carrier wavelength, i.e. with $\bar{\delta}_k = \delta_k/\lambda_c$ and $\Delta\bar{r} = \Delta r/\lambda_c$. As for the gain function $g_{k,i}$ we will use a raised cosine in range, representing the received baseband pulse shape; and we model the one way amplitude gain in azimuth with the normalized sinc function such that

$$g_{k,i} = g_\varphi^2(\varphi_k) g_\nu\left(\nu_{k,i}\right), \quad (8)$$

where

$$\nu_{k,i} = \frac{\delta_k - r_i}{\Delta r} = \frac{t_k - t_i}{\tau'} \quad (9)$$

$$g_\varphi(\varphi) = \begin{cases} \text{sinc}\left(\hat{\varphi}\frac{0.8858}{BW}\right) \\ \hat{\varphi} = (\varphi + 180^\circ \mod 360^\circ) - 180^\circ \end{cases} \quad (10)$$

$$g_\nu(\nu) = \begin{cases} \frac{1}{2}\left[\cos\left(2\pi\nu - \pi\right) + 1\right], & \text{if } 0 \leq \nu \leq 1. \\ 0, & \text{otherwise,} \end{cases} \quad (11)$$

where the angle φ has unit degrees and BW is the one way antenna beam width. Letting $\nu = t/\tau'$ for $0 < t < \tau'$ we get the frequency response of $g_\nu(\nu)$ as

$$|G_\nu(f)|^2 = \frac{\tau'^2}{4}\text{sinc}^2\left(\frac{\tau' f}{2}\right)\left(\frac{\cos\left(\frac{\pi\tau' f}{2}\right)}{1-f^2\tau'^2}\right)^2 \quad (12)$$

with 3 dB bandwidth of $1.44/\tau'$. Provided that $M \geq N$, the total bandwidth over all M pulses can be approximated as

$$B \approx \frac{N-1+1.44}{\tau'}. \quad (13)$$

Finally, we need a method of generating the point scatterers strength and position, s_k, δ_k, and α_k. Accurately modeling specific targets is by no means a trivial task; as an example, see [17]. Here we will instead model targets as arbitrary 2D clouds of point scatterers, of which a standard Rayleigh target can be considered a special case, [7, p. 75-78]. We want to train the neural network to increase the range resolution. Since we should not expect the resolution to be greater than (1), we should generate samples with at least this amount of spacing between targets. For B in (13) this yields a spacing between targets of at least $d_{min} \approx \Delta\bar{r}/N$.

Let the total range window of interest be $[\bar{r}_0, \bar{r}_0 + \Delta\bar{R}]$, where $\Delta\bar{R} = \Delta R/\lambda_c \geq \Delta\bar{r}$. The chosen procedure for generating s_k, δ_k, and α_k is then as follows:

1) Draw an integer number of targets $N_T \in [1, N_t^{max}]$ with uniform probability.
2) Draw N_T slots, d_j, $j = 1, ..., N_T$, from the uniform distribution $\mathcal{U}(0,1)$ and normalize and scale such that $\sum_{j=1}^{N_T} d_j = \Delta\bar{R}$, and ensure that the smallest d_j is at least two d_{min} wide.
3) For each target, draw a target width $w_j \sim \mathcal{U}(0, d_j - d_{min})$ and then draw a target position within slot $p_j \sim \mathcal{U}(d_{min}/2 + w_j/2, d_j - w_j/2 - d_{min}/2)$.
4) For each target, draw an integer number of scatterers $\kappa_j \in [1, \kappa^{max}]$ with uniform probability. The total number of scatterers is then $K = \sum_{j=1}^{N_T} \kappa_j$.
5) For each target, draw scatterer positions within width w_j, $\delta''_{j,i} \sim \mathcal{U}(-w_j/2, w_j/2)$ for $i = 1, ..., k_j$. The total distance to the i'th scatterer in the j'th target is then $\delta'_{j,i} = p_j + \delta''_{j,i} + \sum_{t=1}^{j-1} d_t$. We then relabel all $\delta'_{j,i}$ with indices $k = 1, ..., K$ and let $\delta_k = \bar{r}_0 + \delta'_k$.
6) For each scatterer, draw an angle $\alpha_k \sim \mathcal{U}(-BW/2, BW/2)$.
7) For each target, draw an average strength $\bar{s}_j \sim \mathcal{U}(\bar{s}_{min}, \bar{s}_{max})$, and for each scatterer draw a strength $s'_k \sim \mathcal{U}(s'_{min}, s'_{max})$ and let $s_k = \bar{s}_j s'_k \eta_k$, where $\eta_k = \frac{1}{(1+\delta'_k/\bar{r}_0)^2}$ is a normalized range dependent amplitude gain.

The chosen values of all parameters presented in steps 1) - 7) are shown in Table I. This completes our model for generating a signal

$$\begin{aligned}\mathbf{x}_i &= [|v_{0,i}|^2, ..., |v_{M-1,i}|^2]^T \\ &= [x_{0,i}, ..., x_{M-1,i}]^T. \end{aligned} \quad (14)$$

B. The Label

We now consider the required label vector $\mathbf{y}_i$. We formulate a representation of all scatterers as an N-point vector $\mathbf{y}_i$ with n'th element given by

$$y_{n,i} = \sum_{k=0}^{K-1} \frac{(s_k g_{k,i})^2}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(n + \frac{1}{2} - \nu_{k,i}N\right)^2}{2\sigma^2}\right) \quad (15)$$

for $n = 0, ..., N-1$. This Gaussian kernel filtered version of the scattering points will have a resolution that depends on σ^2. With $\nu = t/\tau'$, the 3 dB bandwidth of the Gaussian kernel is

$$B_G = \frac{\sqrt{\ln(2)}}{\pi\sigma}\frac{N}{\tau'}. \quad (16)$$

By setting $\sigma^2 = \ln(2)/\pi^2$ we set the kernel bandwidth approximately equal to the full bandwidth in (13), which gives the theoretical range resolution in (1).

Another choice for $\mathbf{y}_i$ could be the result obtained from a coherent system, i.e. when the phase ϕ_m in (2) is known for each pulse. This gives a trivial solution to the proposed problem. Assume all scatterers are confined to an interval of one pulse length such that

$$\delta_k = r_i + \frac{n + \Delta n_p}{N}\Delta r \quad (17)$$

for some integer $n \in [0, N-1]$ and Δn_p is uniformly distributed in the interval $(0, 1]$. We then re-label all s_k with index n, p, i.e. $s_k \to s_{n,p}$ for $p = 0, ..., P_n - 1$, where P_n is the number of scatterers in the interval $[n, n+1]$. Ignoring noise, we can then express (4) as

$$v_{m,i} \propto \beta_m e^{-j\theta_{m,i}} \sum_{n=0}^{N-1} \tilde{y}_{n,m,i} e^{-j2\pi(m-m_0)\frac{N-1}{M-1}\frac{n}{N}} \quad (18)$$

where

$$\tilde{y}_{n,m,i} = \sum_{p=0}^{P_n-1} s_{n,p} g_{n,p,i} e^{-j2\pi(m-m_0)\frac{N-1}{M-1}\frac{\Delta n_p}{N}} \quad (19)$$

$$\theta_{m,i} = 2\pi\left(\frac{m(N-1)}{M-1} - m_0\right)\frac{N-1}{M-1}\frac{r_i}{\Delta r} - \phi_m \quad (20)$$

$$m_0 = \frac{M-1}{2} - \frac{2\Delta r(M-1)}{\lambda_c(N-1)}. \quad (21)$$

For $M = N$ and using (14) we can write (18) as

$$\mathbf{x}_i \propto \left|\text{diag}\left(F_{m_0}\tilde{Y}_i\right)\right|^2 \quad (22)$$

where F_{m_0} is the N-point DFT matrix shifted in frequency by m_0 and $\tilde{Y}_i$ is an $N \times N$ matrix with elements $\tilde{Y}_{n,m,i} = \beta_m e^{-j\theta_{m,i}}\tilde{y}_{n,m,i}$. We can thus approximate $\mathbf{x}_i$ as the squared magnitude of the N-point DFT of $\tilde{\mathbf{y}}_i = \tilde{y}_{0,m,i}, ..., \tilde{y}_{N-1,m,i}$ rotated by $\theta_{m,i}$ and modulated by m_0.

Regarding the elements of $\tilde{\mathbf{y}}_i$, we note that these depend on the frequency index m, and by taking the IDFT of (18) we will get an approximation of $\tilde{\mathbf{y}}_i$ with undesired artifacts. This makes it less interesting as a training label. We will however use $|\tilde{\mathbf{y}}_i|^2$ as a reference when evaluating our results. Since $|\tilde{\mathbf{y}}_i|^2$ spans a range $\Delta\bar{r}$ the resolution of $|\tilde{\mathbf{y}}_i|^2$ is $\Delta\bar{r}/N$, an increase in resolution of a factor $N/1.44$ due to the cosine shape of the pulse. To distinguish between range resolution

before and after pulse compression we will use the term "synthetic range cells" after pulse compression and "range cells" before pulse compression. Note that over several range cells $i = 1, ..., N_r$ the expression in (18) can be approximated as a short-time-Fourier-transform (STFT) where the pulse shape acts as the window function.

As the aspect angle changes, the radial positions of all scatterers will change slightly giving rise to the well known glittering effect, or target fluctuations, often modelled with one of four Swirling models (Rayleigh Targets) [18].

C. Ambiguities and SNR

The uniqueness of an inverse phase problem applies up to so called $trivial\ ambiguities$ [9], since we can always write

$$|\mathbf{G}(f)| = |\mathcal{F}\{g(t)\}| = |\mathcal{F}\{g(t-\tau_0)\}| = |\mathcal{F}\{g(-t)\}|.$$

We can therefore not expect the network to learn the correct shift or reflection of $\mathbf{y}_i$. This issue is encountered in any phase retrieval problem and the only way to solve it without prior knowledge of $\mathbf{y}_i$, is to correlate or match overlapping estimates $\hat{\mathbf{y}}_i$ and $\hat{\mathbf{y}}_{i+1}$ where we have sampled the range with period $0 < T_s < \tau'$.

This is exploited in the iterative Griffin-Lim algorithm for phase retrieval [19], which is based on the STFT. The known overlap of samples and window shape acts as the only prior knowledge. If additional prior knowledge is known a wide range of phase retrieval algorithms exist for the STFT [20]. We look closer at the Griffin-Lim algorithm in Section III-D. As with the Griffin-Lim algorithm, we will feed the network with N_r consecutive range samples of $x_{m,i}$, for $i = 0, ..., N_r - 1$, at each pulse m, yielding a STFT "block", and leave the process of exploiting overlapping information to the network. The range sampling is such that

$$r_i = r_0 + i\varepsilon\Delta\bar{r} \tag{23}$$

where $\varepsilon = T_s/\tau' \in [0, 1]$. Our input $\mathbf{X}$ will then be an $M \times N_r$ matrix

$$\mathbf{X} = [\mathbf{x}_0, ..., \mathbf{x}_{N_r-1}] \tag{24}$$

covering a range

$$r_X = \Delta\bar{r}(1 + (N_r - 1)\varepsilon). \tag{25}$$

Each element of our label $\mathbf{y}$ is now

$$y_n = \frac{1}{N_r} \sum_{i=0}^{N_r-1} y_{n,i} \tag{26}$$

for $n = 0, ..., N'$ where $N' = N(1 + (N_r - 1)\varepsilon)$. For practical reasons we will choose N and ε such that $N\varepsilon$ is an integer.

Also, as explained in Section II-D we have set $M = 2N$, and as stated in [21], the inverse problem is then equivalent to recovering $\mathbf{y}$ from its autocorrelation function, i.e. any two $\mathbf{y}_1$ and $\mathbf{y}_2$ with the same autocorrelation function will be indistinguishable.

Regarding the SNR we only consider the range cell with the highest power when setting the SNR of a target, which may or may not span several range cells. The SNR is thus defined as

$$\text{SNR} = \max_i \frac{1}{M} \frac{\sum_{m=0}^{M-1} \tilde{x}_{m,i}}{2\sigma_w^2} \tag{27}$$

where $\tilde{x}_{m,i}$ is the noise free version of $x_{m,i}$.

D. Network model

As is often the case with neural networks, the input data $\mathbf{X}$ and label data $\mathbf{y}$ are first normalized in order to reduce the required dynamic range of the network and ease the training process. Here, this normalization also includes logarithmic scaling and is done by setting

$$\bar{p} = \frac{1}{N_r} \sum_{i=0}^{N_r-1} \sum_{m=0}^{M-1} x_{m,i} \tag{28}$$

$$x'_{m,i} = \ln\left(\frac{M x_{m,i}}{\bar{p}}\right) \tag{29}$$

for $m = 0, ..., M - 1$ giving $\mathbf{X}' = [\mathbf{x}'_0, ..., \mathbf{x}'_{N_r-1}]$ a mean value of one in linear scale. We also normalize the label in the same manner. Since many elements of $\mathbf{y}$ may have value zero, we need to add a small bias and therefore set

$$y'_n = \ln\left(\left(\frac{N' y_n}{\sum_{n=0}^{N'-1} y_n} + 0.001\right)\frac{1}{1.001}\right) \tag{30}$$

giving $\mathbf{y}'$ a mean value of one in linear scale. The choice of this bias affects the network performance, and a different value may be more suitable for a different signal model. Our final estimate, $\hat{\mathbf{y}}$, is then obtained as

$$\hat{\mathbf{y}} = \frac{\bar{p}}{N'}\left(1.001\exp(\hat{\mathbf{y}}') - 0.001\right) \tag{31}$$

where $\hat{\mathbf{y}}'$ is the DNN output of length N'. The gain in SNR after the DNN can be estimated using (26) and (15). A sample $\mathbf{X}$ containing only noise will have a randomness consistent with a large target with many point scatterers. After normalization these two will have similar patterns. We can thus approximate the mean output of the DNN for a noise only sample by making (15) continuous in ν_k and setting s_k to some arbitrary constant, e.g.

$$y_{n,i}^{\text{noise}} = \int_{-\infty}^{\infty} \frac{g_\nu^2(\nu - \tilde{r}_i)}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(n + \frac{1}{2} - (\nu - \tilde{r}_i)N\right)^2}{2\sigma^2}\right) d\nu \tag{32}$$

where we have ignored the azimuth gain and $\tilde{r}_i = r_i/\Delta r$. We then use (26) to obtain y_n^{noise} from (32) and get the noise power as

$$\bar{P}_{\text{noise}} = \frac{y_{N'/2}^{\text{noise}}}{\sum_{n=0}^{N'-1} y_n^{\text{noise}}} \tag{33}$$

where we estimate the noise power as the peak, or mid point, of the DNN output. Note that this assumes that the DNN output is converted to linear scale. We can do the same calculation for a single point scatterer and take the ratio of the peak of the DNN output and $\bar{P}_{\text{noise}}$ to get the gain in SNR. With the chosen parameter values this results in a gain of 13 dB. For low SNRs this approximation will not hold as the

noise will result in a widening of the synthetic pulse shape, causing decreased gain in SNR due to the normalization in (28) and (30). This also assumes a good fit to the label after training, and therefore this gain in SNR is only approximate.

The DNN consists of two parts. The first part has four convolutional layers where each of the N_r columns in the input are transformed to a 16 element vector by the same set of weights. The second part has four fully connected layers transforming the $N_r \times 16$ data points to the $N' \times 1$ estimate $\hat{\mathbf{y}}'$. The motivation for this setup follows from the reasoning in Section II-C, i.e. in the first part we transform the data to time domain and in the second we correlate and align the N_r samples. This is at least the motivation for it but whether this is what is actually happening or not is a different question.

According to [22] the number of frequency samples required for phase retrieval is $M \geq 2N-1$ and we therefore set $M = 2N$. This applies to real valued $\mathbf{y}$, and as mentioned $\mathbf{y}$ is complex in this case, for which the required M is actually $M \geq 4N-2$. Since we are only interested in the magnitude of $\mathbf{y}$ we still choose $M = 2N$. Experiments were performed with $M = 4N$ and gave comparable results as $M = 2N$. Experiments were also performed with $M = N$ and gave worse performance than $M = 2N$, see Section III-A.

As for ε and the number of range samples in $\mathbf{X}$, N_r, a grid search was performed over a range of values. As could be expected the performance improved with larger N_r and a value of $N_r = 5$ was finally chosen. With sampling interval $\varepsilon = 0$ we get a repetition of the same samples and gain no information. With $\varepsilon = \tau'$ we have no overlap of information in two consecutive range cells. It is therefore not surprising to find an optimal value within this range, which for the chosen setup was $\varepsilon = 4/32$.

From our spacing between targets constraint, d_{min}, we are restricted to set $N_t^{max} \leq N/2$. However for $N_t^{max} = N/2 = 16$ we get deterministic positioning of targets which is in this case undesired as we do not want the network to be biased to any particular target configuration. For this reason the maximum number of targets was limited to 10. All other parameter values were arbitrarily chosen. All distances are normalized to wavelengths. As an example, for an X-band radar at $f_c = 10$ GHz these numbers would correspond to a system with full pulse length Δr = 200 m, 24 MHz bandwidth and 25 m interval in range sampling.

The final network architecture is presented in Table II. The values of the kernel sizes and number of layers are a result of empirical testing, as is often the case with neural networks. A deeper network did yield slightly lower validation loss at the expense of longer execution time. Since a motivating factor to choose a DNN based solution is the relatively fast and deterministic computation time, as DNNs can be evaluated efficiently by parallel computing, we aimed for a small network that still gave satisfying results. The purpose of this study is mainly to demonstrate that this type of non-coherent pulse compression with DNNs is feasible, rather than finding the optimal such solution. The gain beyond the eight layers was subtle with a training/validation loss of 6.36/6.36 for 8 layers and 6.33/6.33 when adding one additional fully connected layer of the same size. Subtracting one of the four fully connected layers gave a training/validation loss of 6.43/6.43, trained on 10^6 samples for 10 epochs. Network architecture optimization is of course more complicated than simply adding layers. As with depth, a wider network will in general yield lower loss at the expense of computation time. As an example, increasing the output dimension of the first three fully connected layers to 100 instead of 48 gave a validation loss of 6.06 instead of 6.36, however this also entails an increase in parameters by over 80%. The chosen filter length of 16 in the first convolutional layer has been optimized with regard to the chosen distribution of SNRs. If only high SNRs were considered a longer filter tended to yield lower loss. The loss was the empirical mean squared error loss

$$L = \frac{1}{N'} \sum_{n=1}^{N'} (\hat{y}'_n - y'_n)^2 \tag{34}$$

and was minimized using stochastic gradient descent with adaptive moment estimation (Adam) [23].

TABLE I
PARAMETER VALUES

Parameter	**Description**	**Value**
$\Delta\bar{r}$	Pulse length in meters normalized by carrier wavelength	6700
M	Number of pulses	64
N	Samples within pulse in neural network output	32
B	Total bandwidth, determined by N and $\Delta\bar{r}$	$0.0024 f_c$
SNR	Signal to noise ratio	-3 to 30 dB
ε	Range sampling interval in pulse lengths	1/8
N_r	Range samples in neural network input	5
N'	Number of synthetic range samples in neural network output, $\mathbf{y}$, determined by N, N_r and ε	48
N_t^{max}	Maximum number targets	10
κ^{max}	Maximum number of scatterers per target	100
$[\bar{s}_{min}, \bar{s}_{max}]$	Interval of "average" strength per target	[0.1, 1]
$[s'_{min}, s'_{max}]$	Interval of initial strength per scatterer	[0.1, 10]
BW	Antenna 3 dB beam width one way/two way	3° / 2.2°

TABLE II
NEURAL NETWORK ARCHITECTURE

Layer type	Weight Shape	Output Shape	activation
Convolutional	F: 64, K: 16×1	$64 \times 49 \times 5$	ReLu
Convolutional	F: 32, K: 1×1	$32 \times 49 \times 5$	ReLu
Convolutional	F: 16, K: 1×1	$16 \times 49 \times 5$	ReLu
Convolutional	F: 16, K: 49×1	$16 \times 1 \times 5$	ReLu
Fully Connected	48×80	48×1	ReLu
Fully Connected	48×48	48×1	ReLu
Fully Connected	48×48	48×1	ReLu
Fully Connected	48×48	48×1	Linear

F: Number of filters/kernels. K: Kernel size. All Convolutional layers had stride [1,1] and no zero padding. Total number of parameters: 27200.

TABLE III
BATCH SIZE AND LOSS

Batch Size	Training Loss*	Validation Loss*	execution time†
32	6.20	6.35	14 s
64	6.23	6.38	8 s
128	6.27	6.38	4 s
256	6.45	6.52	3 s
512	6.46	6.54	2 s

*After 100 epochs with 10^5 training samples and 10^4 validation samples. †Time per epoch.

TABLE IV
NUMBER OF PULSES M AND LOSS

M	Training Loss*	Validation Loss*	Network Size†
32	6.54	6.64	F:[64,32,33,20] K:[16×1, 1×1, 1×1, 17×1] $\#W$: 27401
64	6.27	6.38	F:[64,32,16,16] K:[16×1, 1×1, 1×1, 49×1] $\#W$: 27200
128	6.22	6.34	F:[16,10,10,16] K:[32×1, 1×1, 1×1, 97×1] $\#W$: 27288
63⋆	6.35	6.42	F:[64,32,16,16] K:[16×1, 1×1, 1×1, 48×1] $\#W$: 26944

*After 100 epochs with 10^5 training samples and 10^4 validation samples. †F=Number of filters/kernels, K=Kernel sizes in the first 4 convolutional layers. $\#W$=number of weights. ⋆Data was transformed with IDFT.

III. RESULTS

Results on training the DNN are presented in Section III-A. As mentioned in the introduction, pulse compression is used to increase range resolution and SNR. Results on range resolution are presented in Section III-B and results on SNR gain are presented in Section III-C. Although we have limited this study to stationary scenarios, we also present some results on non-stationary cases as well as a comparison to other methods in Section III-D.

A. Training the DNN

In addition to network architecture, the training of a DNN also involves some trial and error when selecting hyperparameters, training time, number of samples, optimization method etc. To simplify this process the parameters of the Adam optimizer were kept to the default values in Keras [24] with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $\eta = 0.001$, and no other optimization methods were considered. Five different batch sizes were tested, the results of which are shown in Table III. Based on this a batch size of 128 was chosen since it gave a good trade-off between execution time, overfitting, and low loss.

Fig. 1 shows the loss as a function of batch iterations for four different sizes of training data. The same validation set with 10^4 samples was used in all four cases. All data sets are subsets of any larger data set, where 10^6 samples is the full data set. Significant overfitting can be seen when the number of training samples is 10^4 and virtually no overfitting occurs with 10^6 samples. A sample size of 10^5 and 100 epochs of training was chosen as this is sufficient four our purposes since the validation loss reaches the same level when using 10^6 samples and decreases negligibly with further training. Note that this number of samples may not be sufficient for a different network. A larger network will typically require more training data to avoid overfitting. The network was trained using Keras [24] with Tensorflow [25] as backend and an Nvidia Geforce RTX 2080 GPU, resulting in a computational time of 4 seconds per epoch.

In order to make the network more robust to noise, the SNR of the training and validation data was uniformly distributed between -3 and 30 dB in logarithmic scale. The mean loss depends on this distribution, and there is a slight trade off between minimizing loss and SNR robustness. A network trained only on data with 30 dB SNR achieved a validation loss of 5.4, and 10.5 when evaluated on data with 3 dB SNR. Conversely, a network trained only on data with 3 dB SNR converged to a validation loss of 7.4, and 16.5 when evaluated on data with 30 dB SNR. A network trained on the proposed range of SNRs gave a loss of 7.5 on data with 3 dB SNR and 5.6 on data with 30 dB SNR, see Fig. 2. The networks performance at a specific SNR does therefore not appear to suffer too much in terms of loss despite having been trained on a broader distribution of SNRs. As previously mentioned we make no claim to have selected the optimal values of the chosen parameters but instead aim to present a solution that yields results sufficient to show that this type of solution to non-coherent pulse compression is feasible. Since all activations were rectified linear units (ReLu), all weights were initialized with He initialization [26].

Tests were performed with $M = N$, $M = 2N$ and $M = 4N$, where $N = 32$ is the number of synthetic resolution cells within one pulse after pulse compression. The number of filters and kernel sizes in the first four convolutional layers were modified such that the total numbers of parameters were comparable and the output shape from the last convolutional layer was always 16×1×5. The loss for $M = 128$ was comparable to the loss when $M = 64$ while the loss for $M = 32$ was slightly higher, see Table IV. For this reason we settled with $M = 64$.

Tests were also made with transformations of the input data $\mathbf{X}$. If $M = 2N-1$, then $\mathbf{x}_i$ can be regarded as the DFT of the autocorrelation of $\tilde{\mathbf{y}}_i$. One could then take the IDFT of each $\mathbf{x}_i$ to obtain the autocorrelations of $\tilde{\mathbf{y}}_i$ for $i = 0, ..., N_r - 1$. The problem is then shifted from a phase retrieval problem to that of finding the inverse of the autocorrelation function. Tests were run where this was used as input data instead, using the normalization in (28) and (29), and applying the IDFT transform before converting to log scale, with the motivation that this might be an easier problem for the DNN. The loss, shown in Table IV, was similar to that of having no DFT on $\mathbf{X}$, i.e. transforming the data in this way did not yield a better solution.

The average training loss for the chosen solution was 6.3 and the average validation loss was 6.4. The validation loss

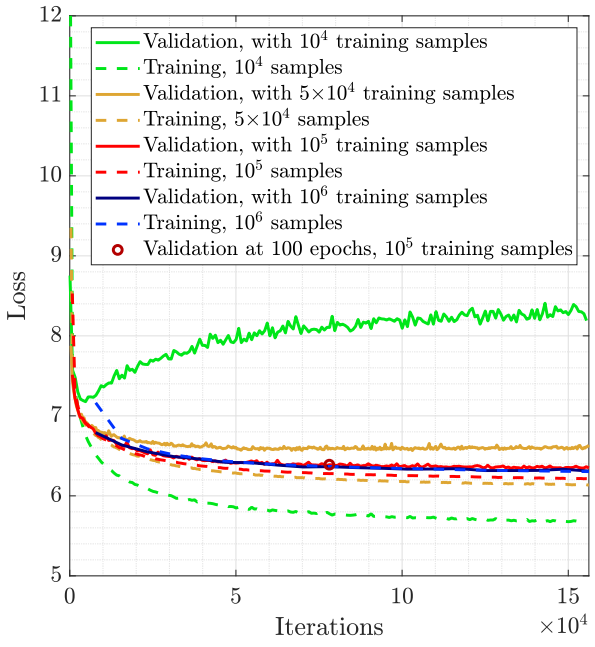

Fig. 1. Training and validation loss for different sizes of training data. The network was trained on each data set an equal amount of time with a batch size of 128. This corresponds to 2000 epochs for 10^4 samples, 400 epochs for 5×10^4 samples, 200 epochs for 10^5 samples and 20 epochs for 10^6 samples. The red circle marks the validation loss (6.4) for the selected training process with 10^5 training samples and 100 epochs training time. The same validation data set with 10^4 samples was used in all four cases.

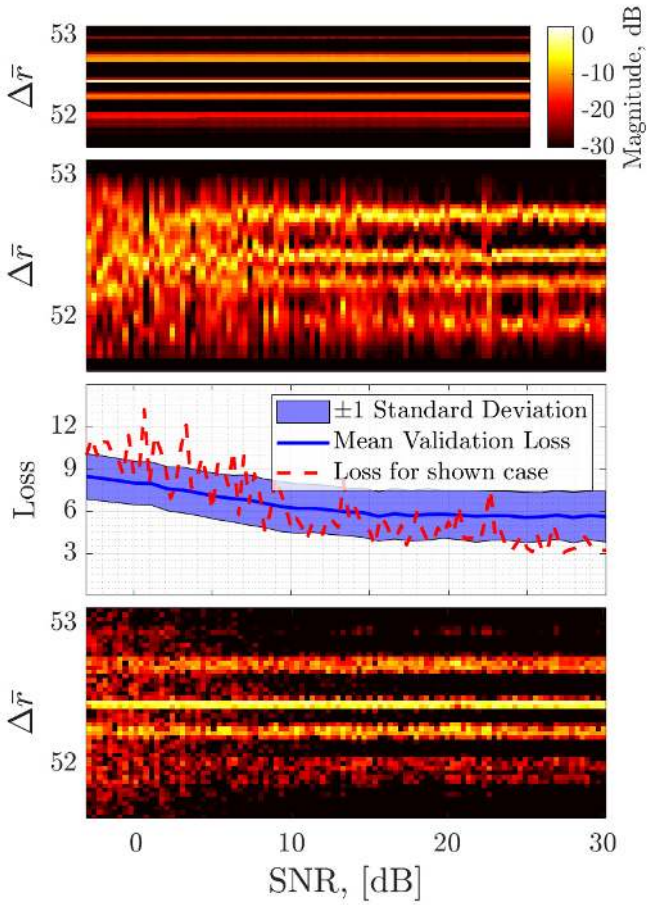

Fig. 2. Example of improvement with SNR for a five target scenario. At each SNR the radar position was also jittered by drawing each cartesian coordinate from a $\mathcal{N}(0, 30^2)$ distribution. This gives a mean deviation of approximately 38 wavelengths from point (0,0) (or 1.1 m with f_c =10GHz). The range to the first cell is 52 $\Delta\bar{r}$ (10.4 km with f_c =10GHz) The top plot shows the ground truth, virtually the same for all SNRs. The second is the DNN estimate as a function of SNR. The variation at high SNRs (>20dB) is mainly caused by the random radar position. The third plot shows the loss at each SNR for both this specific case and the mean validation loss ±1 standard deviation. The bottom plot is the coherent IDFT estimate, shown as reference. The color axis scale is the same in the first, third, and fourth plot. Below 10 dB SNR the performance of the DNN solutions drops significantly in the second plot which correlates well with the loss in the third plot.

as a function of SNR is shown in the third plot in Fig. 2. Since the loss is taken on $\mathbf{y}'$ in logarithmic scale, it has little physical meaning. As reference, the MSE solution of the linear transform $\mathbf{y}' = \mathbf{W}\mathbf{z}$ with vectorized $\text{vec}(\mathbf{X}') = \mathbf{z}$, had average loss $L \approx 10$ for both training and validation data. We note that the loss increases for SNR$<$ 10 dB.

To further visualize this, Fig. 2 shows a case with five targets where we have added the coherent estimate as reference. In addition to varying the SNR we have also jittered the radar position by drawing each cartesian coordinate from a $\mathcal{N}(0, 30^2)$ distribution. This gives a mean deviation of $\sigma\sqrt{2\pi}/2 \approx 38$ wavelengths from point (0,0). The range to the first cell is 52 $\Delta\bar{r}$. This gives rise to "glittering" or fluctuations in magnitude between samples, the effect of which is what we see at higher SNRs in Fig. 2 in both coherent and non-coherent estimates. At SNR lower than 10 dB the distortion is mainly caused by noise.

B. Resolution Gain

With linear systems it is common to specify the resolution of a pulsed compressed radar in terms of the beamwidth and peak-to-sidelobe ratio of the synthetic pulse, i.e. the pulse shape after pulse compression. In this case however we have a highly non-linear neural network functioning as the pulse compression "filter", and conclusions drawn from such metrics may not apply in the same way as in linear systems. Instead we are left to numerical evaluations where we evaluate the DNN over a broad range of simulated scenarios.

We begin with what can be regarded as a synthetic pulse shape by looking at the return of a single point scatterer with no noise. This is shown in Fig. 3. As in [27] we will benchmark the performance against an equivalent coherent system. The coherent synthetic pulse shape is also shown in Fig. 3 with and without windowing before the DFT. Without windowing we get the sinc-pattern with 13.3 dB sidelobes. For our comparison we decided to use a windowed version that gave a synthetic pulse shape that was more similar to the one obtained with the DNN. Although the DNN pulse shape does not have sidelobes in the same way it is interesting to note that there are similar patterns of "peaks" at approximately the same locations as the coherent pulse, at positions 0.5,1.5,...,47.5. These are the result of the network optimization and were not present in the label $\mathbf{y}$. The "sidelobe" levels of the DNN are

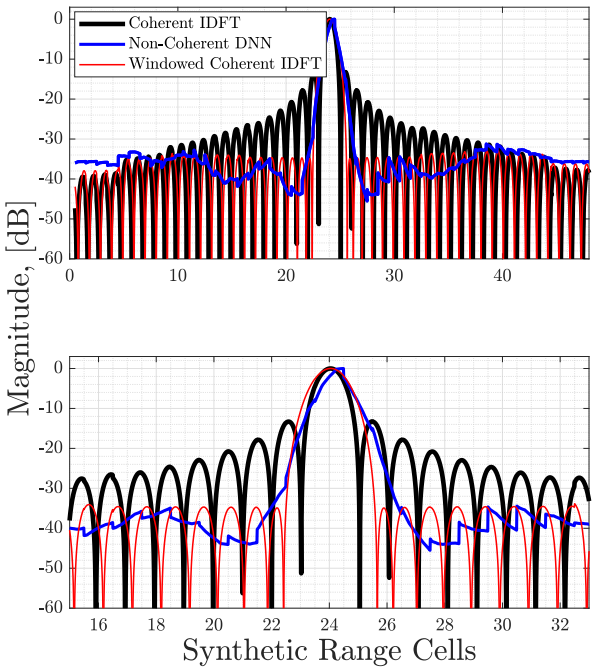

Fig. 3. Numerically evaluated synthetic pulse shape over all $N' = 48$ synthetic range cells. The coherent solution using the IDFT is also shown as reference without windowing, yielding the characteristic sinc pattern, and with a Taylor window with -35 dB sidelobe suppression over 50 sidelobes. The high resolution was obtained by sliding a single point scatterer over a range of ± 0.5 synthetic range cells in 100 steps, centred at the mid cell indexed 24.

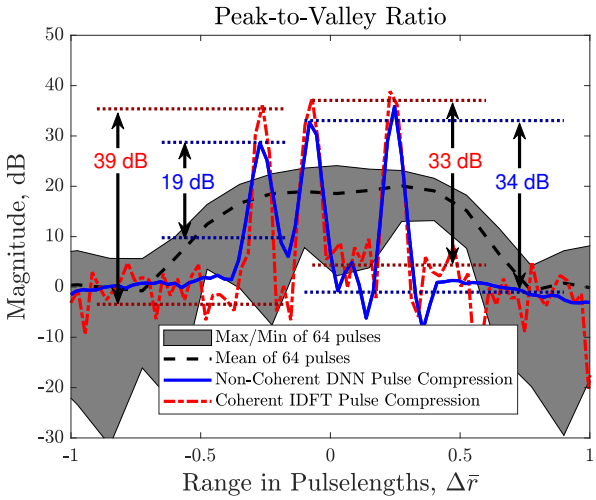

Fig. 4. Example of peak-to-valley ratios for a scenario with three targets. The mean pulse shape before pulse compression is shown for reference as the black dashed line as well as the maximum and minimum values of the 64 pulses. The SNR was 20 dB.

TABLE V
REQUIRED RANGE† SEPARATION FOR ≥3 DB PEAK-TO-VALLEY RATIO WITH **90%** AND 95% EMPIRICAL PROBABILITY

SNR [dB]	Non-Coherent DNN*	Coherent IDFT*
20	**4.8**/5.6	**2.8**/3
15	**5**/5.8	**2.8**/3
10	**6.6**/7.2	**2.8**/3
7	-/-	**2.8**/3
3	-/-	**2.8**/3.4

†Separation in synthetic range cells for two point scatterers. *Bold number corresponds to **90**% and plain number corresponds to 95%

at a rather constant level and below -35 dB. For this reason, a Taylor window was used with -35 dB sidelobe suppression over $\bar{n} = 50$ lobes, [28]. This results in a widening of the -3 dB pulse width by a factor 1.2 which also makes it closer to the apparent shape of the DNN pulse shape. In addition to this it also introduces a SNR loss of 0.9 dB.

The -3 dB pulse width of the DNN pulse was measured to about 1 synthetic range cell, although it quickly gets wider at lower gains. Note that for the coherent case we only need samples from one range cell to get the N-point synthetic range, i.e. one $\mathbf{x}_i$ for some i while the DNN requires $N_r = 5$ consecutive range samples. For the sake of comparison we let the coherent IDFT estimate be the average of $N_r = 5$ overlapping samples of $\mathbf{x}_i$. It thereby uses the same input data, $\mathbf{X}$, as the non-coherent DNN solution and produces the same output size N'.

To quantify the resolution, we use the peak-to-valley ratio which we define as

$$\frac{\min(P_1, P_2)}{V} \tag{35}$$

where P_1 and P_2 are the power levels of two consecutive peaks/targets and V is the "valley" power estimated from the signal power between the most adjacent scatterers of the two targets. This was estimated as the mean of the n smallest synthetic range cells out of a maximum of 5 where $1 \leq n \leq 3$, depending on the distance between peaks. This is illustrated in Fig. 4. Note that the range covered in Fig. 4 is two full pulse lengths, equivalent to 64 synthetic range cells. This was obtained by averaging several estimates with a sliding window approach over range cells with stride equal to one.

In plot (a) in Fig. 5 we estimate the peak-to-valley ratio for two single point scatterers of equal size. The average peak-to-valley ratio over 1000 Monte Carlo runs is shown as a function of distance between targets in synthetic range cells. Table V shows the minimum distance in synthetic range cells required for the peak-to-valley ratio to be at least 3 dB with 90% and 95% empirical probability. For SNRs 3 dB and 7 dB this probability was never reached with the DNN. Note that these incorporate losses due to interference between targets as well as noise. Had it not been for these factors, two point scatterers should be separable with 3 dB margin after a separation of about one synthetic range cell with coherent pulse compression. With the used Taylor window the 3 dB peak-to-valley ratio occurs after a separation of about 1.08 synthetic range cells.

The results in plot (a) in Fig. 5 and Table V are only for two single point scatterers. Similar simulations were performed using samples drawn from the model presented in Section II-A, from which the training and validation data was generated, but with slight modifications. The number of targets generated were between two and ten (not one and ten) and the peak-to-

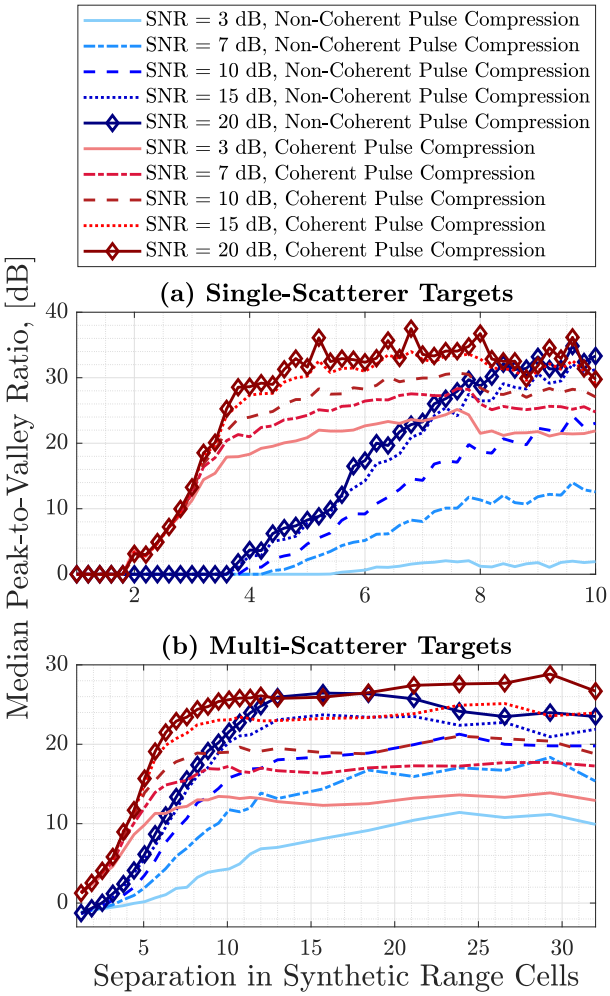

Fig. 5. Empirical median peak-to-valley ratio as a function of the separation between targets in synthetic range for different SNRs. The median was taken from 1000 runs. The distribution of peak-to-valley ratios was quite skewed. For this reason the median was used as the average rather than the mean. **(a)** Two point scatterers of equal size. The peak-to-valley ratio begins increasing after a separation of about two synthetic resolution cells for coherent pulse compression and about four synthetic cells for the non-coherent pulse compression, depending on SNR. **(b)** Multiple targets with multiple scatterers, drawn from a modified version of the model in Section II-A. The modifications included the number of targets to be at least two and the relative peak-to-peak ratio was no more than 10 dB after coherent pulse compression. Although the non-coherent pulse compression requires a larger separation between targets than coherent pulse compression, it can still reach the same levels if the SNR is sufficiently high. Note that the SNR is before pulse compression and includes the power of all targets. The separation in synthetic range refers to the distance between the end and beginning of two consecutive targets.

peak ratio of two consecutive targets defined as

$$|10\log_{10}(P_1/P_2)| \tag{36}$$

using the power levels of the coherent IDFT estimate, was at most 10 dB. Note that the specified SNR is before pulse compression and is thus an average over all targets within one pulse.

The corresponding results for these tests are shown in plot (b) in Fig. 5 and Table VI. Looking at Table V and VI we see that for SNR 10 dB and above, the resolution is approximately a factor 2 worse in synthetic range than the coherent solution using this metric. We also note in plot (b) in Fig. 5 that the median peak-to-valley ratio of the non-coherent DNN solution is approximately at the same level as the coherent peak to valley ratio when SNR$\geq$10 dB and the target separation exceeds 10 synthetic range cells.

TABLE VI
REQUIRED RANGE† SEPARATION FOR $\geq$3 DB PEAK-TO-VALLEY RATIO WITH **90%** AND 95% EMPIRICAL PROBABILITY WITH MULTIPLE TARGETS

SNR [dB]	Non-Coherent DNN*	Coherent IDFT*
20	**7.4**/9.4	**6.2**/7.8
15	**7.8**/9.8	**6.2**/8.2
10	**9.8**/11.8	**6.6**/9.4
7	**11.4**/17.4	**6.6**/9.4
3	-/-	**7**/9.4

†Separation in synthetic range cells for multiple multi-scatter target scenarios.
*Bold number corresponds to 90% and plain number corresponds to 95%

Finally, we show an example where we have combined several estimates of $\mathbf{y}$ to form a scan image or B-scope view. We let the scan rate and PRF be such that we get 64 pulses in 2° and we sample 25 range cells and scan 20°. We then slide a window over the sampled data with size 5×64 and stride [1,64]. For each 5×64 sample we get a 48×1 estimate $\hat{\mathbf{y}}_i$ for $i = 0, ..., 20$. Two estimates $\mathbf{y}_i$ and $\mathbf{y}_{i+1}$ will then overlap by $48 - 4 = 44$ samples and in the final image we simply take the mean of all overlapping samples. This is shown in Fig. 6 for a scenario with three targets within a pulse length. As seen, the three targets can be resolved despite the loss of phase information using the DNN.

C. SNR Gain

In this section we will consider the gain in SNR. Fig. 7 shows the synthetic pulse shape, i.e. simulated echo from a single point scatterer, for various SNRs. The gain in SNR becomes less significant for SNR below 3 dB.

Fig. 8 shows the mean SNR gain for single point scatterers as a function of SNR as well as an intensity map to illustrate the distribution in gain. The empirical 5th percentile for the coherent gain was 19.5 dB for SNR$\geq$0 dB. For the non-coherent gain and SNR$\geq$ 5 dB the 5'th percentile gain was greater than 6 dB and the mean gain was greater than 12 dB, consistent with the 13 dB calculated in Section II-D. In terms of the mean, this is a loss of 10 dB from the coherent case.

In Fig. 6 we can see that the background noise characteristics is different in the coherent and non-coherent estimates. This will affect the trade off between false alarm and detection probability, known as receiver operating characteristics (ROC). This is estimated in Fig. 9 for a target consisting of five point scatterers uniformly confined to one synthetic range cell, each with exponentially distributed RCS. The scatterer positions and RCS were constant in each sample, or set of 64 pulses, but varied between samples. This gave a distribution of the signal power, or target RCS, that was close to that of a target with log-normal probability density function (PDF), a common

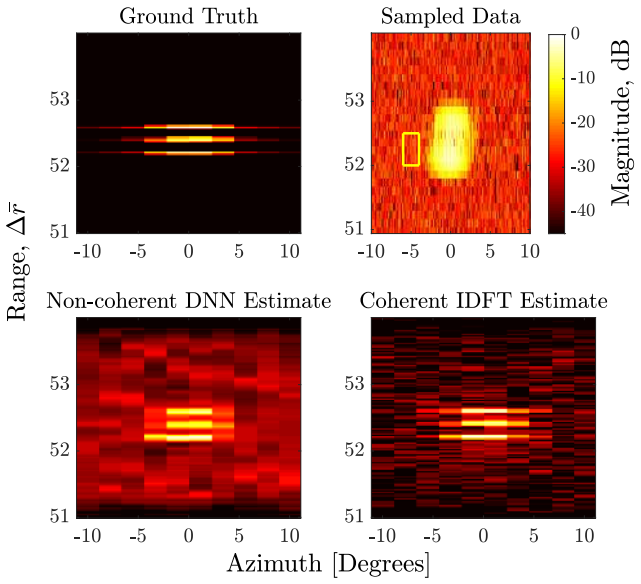

Fig. 6. Example of a simulated scan with three targets positioned within one pulse length $\Delta\bar{r}$, at 52 pulse lengths range. With $f_c = 10$GHz this corresponds to 10.4 km ($\Delta r = 200m$) requiring a pulse repetition frequency < 14kHz. The scan has 64 pulses in 2°, which would then require a scan rate less than $360^\circ/0.83$s for a mechanical antenna system. The SNR in $\mathbf{X}$, top right plot, was 20 dB. The bottom right plot is for reference and shows the coherent result using the IDFT. The yellow box in the top right plot marks the input size to the DNN. The loss at the target for the DNN estimate in the bottom left plot was $L \approx 4$ and $L \approx 14$ in the target free area. The color axis scale is the same in all four images. As seen, the three targets can be resolved despite the loss of phase information.

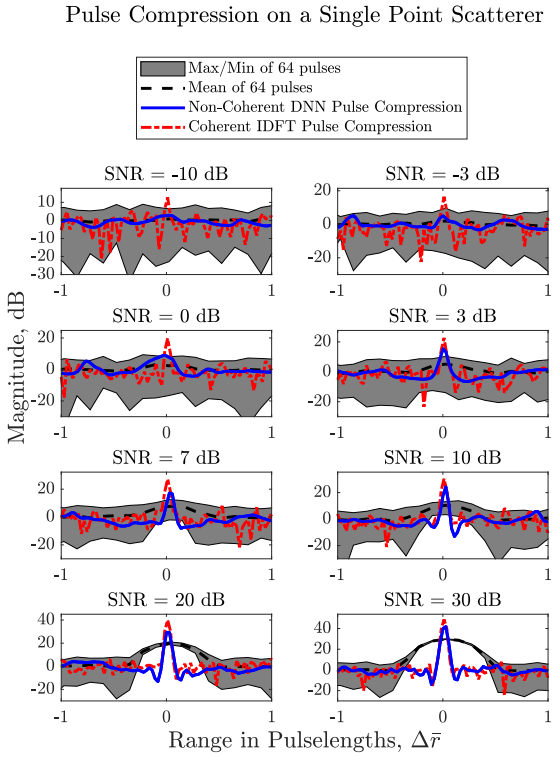

Fig. 7. Example of compressed pulse for different SNRs of a point scatterer. The mean as well as maximum and minimum range of the 64 simulated pulses are shown for reference.

model for target fluctuations from one scan to another. A perhaps more "standard" model is the Swerling III model, [18], which has a similar PDF of target power, see bottom plot in Fig. 9. Fig. 9 shows a comparison of the histogram of 5×10^5 samples of simulated target power and the PDF of a log-normal target scaled to have mass 5×10^5. The ROC estimate is for a constant threshold detector.

In the top plot in Fig. 9 we have compared the DNN solution to an alternative form of non-coherent pulse compression using OOK. We have compared it to a waveform using m-sequence coding of sub-pulses, see [29], with the same instantaneous bandwidth as the DNN solution. The length of such an m-sequence is $2^m - 1$ which gives 63 sub-pulses for $m = 6$. After Manchester coding, [29], the total length of the pulse sequence is 126 sub-pulses with 63 "on" pulses, i.e. it uses approximately the same number of pulses as the presented DNN solution but transmitted within one PRI instead of 64 with a single "sub-pulse" in each. We see in Fig. 9 that simply integrating 64 pulses yields better results in terms of ROC than the other non-coherent solutions. The DNN solution yields equal or better results than the 6-sequence OOK waveform for SNR greater than 7 dB. A shorter m-sequence will have worse performance, and a longer better in terms of ROC. Note that the 6-sequence OOK solution is obtained from a single pulse and the DNN solution requires 64. On the other hand, the 6-sequence OOK solution offers no increase in resolution which is the main objective in this study. Also note that unless a radar can receive and transmit simultaneously the blind range of such a 6-sequence OOK system is 126 sub-pulses, or ca 25 km if Δr=200m. The coherent estimate had empirical detection probability of approximately one, with false alarm probability of almost zero for the three considered SNRs.

D. Other Scenarios and Methods

We saw in Section II-C that the problem presented in this study could potentially be solved using certain phase retrieval algorithms. Fig. 10 shows a comparison between the presented DNN solution and an alternative phase retrieval solution using the Griffin-Lim Algorithm (GLA) [19]. This exploits the fact that the N_r columns of $\mathbf{X}$ can be regarded as the STFT of the desired signal $\mathbf{y}$. The known pulse shape and sampling interval then acts as prior knowledge necessary for phase retrieval, although it is not always sufficient [20]. By iterating between time and frequency domain and applying what is known in each domain the error between the estimated $\mathbf{y}$ and true $\mathbf{y}$ will reduce with each iteration, [19]. In this case we stopped the iteration when this error, measured as the mean square error, changed by less than 10^{-6}. This required 100-600 iterations for the five cases shown in Fig. 10, where each case is identical apart from having different pulse shapes.

The pulse shapes are raised cosine filters with different shape parameters $\beta \in [0, 1]$. A value of $\beta = 1$ yields the simple raised cosine in (11). We note that the GLA works reasonably well with rectangular pulses, ignoring the false targets. However when introducing some bandwidth limitations

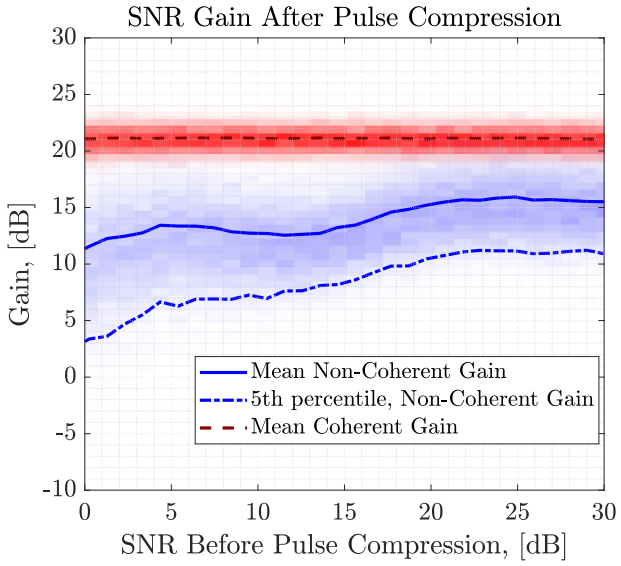

Fig. 8. Empirical mean gain in SNR as a function of SNR for non-coherent and coherent pulse compression. The coherent pulse gain is a result of both pulse compression as well as the averaging of 5 samples. The SNR after pulse compression was estimated as the power at the target divided by the mean power of 20 noise samples minus one. The sample size was 1000 for each SNR.

on the pulse, i.e. increasing β beyond 0.7, the GLA fails. This type of case specific sensitivity is a well known problem with the GLA. This is discussed in [15] where the authors combine the GLA and a DNN to solve a phase retrieval problem in order to gain more robust and accurate phase estimates. This is however a semi-iterative solution where each iteration involves an evaluation of a five-layered neural network. The number of iterations used in [15] was 10, equivalent to a network with 50 layers. Note in Fig. 10 that the DNN solution performs reasonably well for the whole range of β when N=32, despite having only been trained on the pure raised cosine shape were $\beta = 1$.

Another interesting alternation is changing the distribution of the scatterers RCS in the model presented in Section II-A. The model uses uniformly distributed RCS of each point scatterer. One could argue that some other distribution would be more suitable. Due to the normalization of the DNN input data, we can always carefully select a subset of the training data that will be consistent with any choice of distribution that has been normalized, provided that we have a large enough sample size. One could therefore argue that this choice of uniform distribution covers a wide range of possible scenarios even if only a fraction of them are interesting. The loss in Fig. 2 is however the mean when using the uniform distribution and it is therefore motivated to test the DNN with other distributions on RCS. The mean loss for 10^4 samples where the RCS was drawn from the exponential distribution was 6.30 and 6.37 when drawn from a Rayleigh distribution using the same network that had been trained on uniform RCS. This is very close to the validation loss of 6.38.

In this study we have only considered stationary scenarios up to this point. Although dynamic scenarios are left for future work, we will still evaluate the DNN on a couple examples.

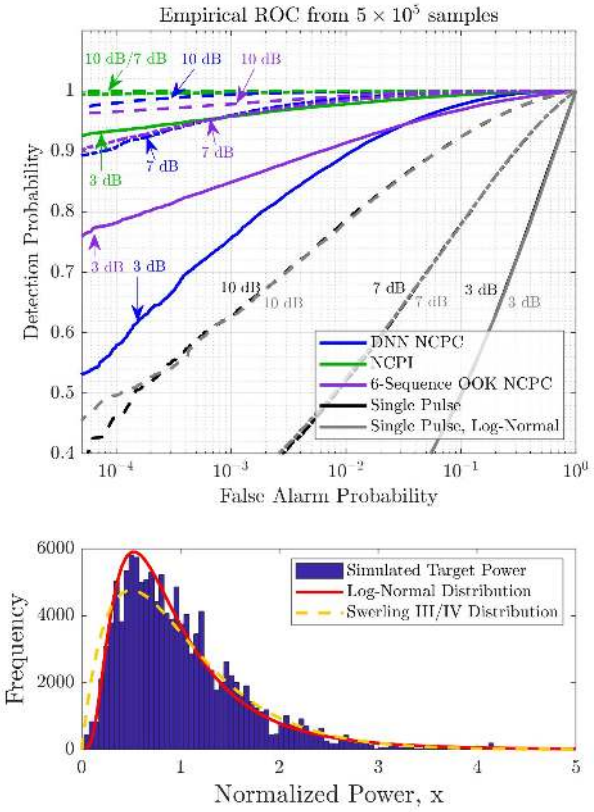

Fig. 9. The top plot shows the ROC curve estimated from 5×10^5 samples. The curves are for the DNN solution (blue), pulse integration over $M = 64$ pulses (green) single pulse detection (black) and a non-coherent pulse compression method with OOK using an m-sequence with m=6 (purple). The m-sequence was generated from a shift register with initial values [1,1,1,1,1,1]. The target consisted of five point scatterers uniformly confined to one synthetic resolution cell, with exponentially distributed RCS. This caused the target power level to have an approximately log-normal like distribution. The bottom plot shows the histogram of the normalized target power levels and the PDF of a log normal distribution and the similar Swerling III/IV PDF scaled to have the same mass. For reference ROC curves for single pulse detection on a true Log-Normal fluctuating target in AWGN is also shown in the top plot (grey). These are quite consistent with the corresponding curves of the simulated target. The DNN ROC curves are for single evaluations of the DNN and no averaging obtained from sliding window estimates, such as the one shown in Fig. 6, was performed. The given SNRs are the average SNRs. NCPC=Non-Coherent Pulse Compression, NCPI=Non-Coherent Pulse Integration.

Relative motion between target and radar is always a problem when using the stepped frequency waveform due to the fact that M pulses are required, [30]. Assume that the target has moved a distance $d = \Delta r L/(N-1)$ pulse lengths along the radar line of sight during the M pulses, typically $L \ll N$. Using $v_{m,i}$ in (4) we can then write the received signal $\bar{v}_{m,i}$ as

$$\bar{v}_{m,i} = \left(\frac{R_0}{R_m}\right)^4 v_{m,i} e^{-j2\pi\alpha_m \frac{mL}{(M-1)(N-1)}} \tag{37}$$

assuming prefect Doppler compensation on the carrier when down converting. The first term is the path loss where R_0 is the range to the target at pulse 0 and R_m is the range at pulse

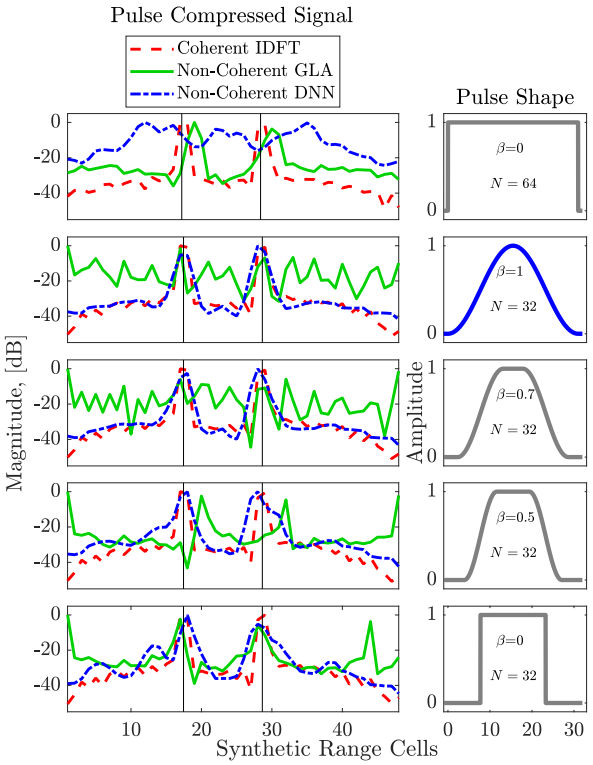

Fig. 10. Comparison between the iterative GLA solution and the presented DNN solution for non-coherent pulse compression. As reference the Coherent solution using the IDFT is also shown. The simulated scenarios show two single point scatterers "illuminated" with different pulse shapes in the form of raised cosine filters. The DNN was only trained on the pure raised cosine with shape factor $\beta = 1$ and N=32. The SNR was 25 dB. The vertical bars mark the true target positions. The number of range samples required for the GLA to work was $N_r = 9$ and the GLA estimate was therefore longer than the $N' = 48$ samples shown here.

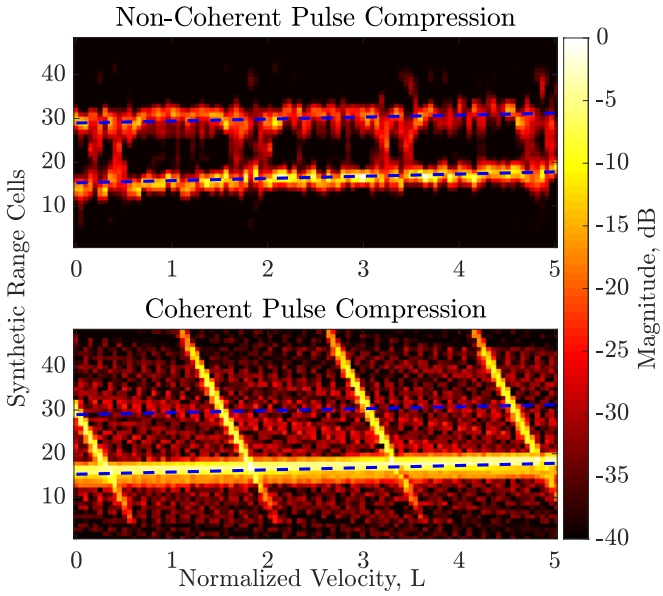

Fig. 11. Scenario with two moving targets where the top target moves with 0.9 times the velocity of the bottom target which is given on the x-axis. The coherent estimate is phase compensated for the velocity of the bottom target. The dashed lines show the true mean position of the targets during $M = 64$ pulses as a function of velocity. The y-axis shows the $N' = 48$ synthetic range cells. As an example, a normalized velocity of one is equivalent to 100 m/s with a 200 m pulse length and 1kHz PRF with N=32 and M=64. The mean SNR was 10 dB.

m. Using (6) the exponent in (37) can be written as

$$-j2\pi \left[\left(\frac{2\Delta\bar{r}L}{(M-1)(N-1)} - \frac{L}{2(M-1)} \right) m + \frac{Lm^2}{(M-1)^2} \right]. \quad (38)$$

The term L is a normalized velocity, i.e., the number of synthetic resolution cells the target has moved during the M pulses, not necessarily an integer. We see that this phase shift contains a linear term in m which will result in a spatial rotation of the estimated signal and a quadratic term in m which will result in a widening of the target.

Theoretically this can be compensated for by multiplying with the inverse phase, but this requires accurate knowledge of the relative velocity between target and radar. This is not a trivial problem, see [30], and [31]. The non-coherent solution is however unaffected by phase shifts. It gets more complicated when the pulse contains several targets that move with different velocities, as this will also influence the signal amplitude levels due to interference caused by the varying distances between targets. Fig. 11 shows such a scenario with two targets consisting of 50 point scatterers with exponentially distributed RCS, uniformly confined to five synthetic range cells where one target moves with 0.9 times the velocity of the other, specified on the x-axis. The coherent solution is only phase compensated for the bottom target, i.e. the velocity of that target is assumed known. Compared to the coherent estimate the non-coherent estimate is relatively unaffected despite not having been trained on such scenarios and does not require any knowledge of the relative velocity between target and radar.

An even more complicated case is when the target rotates, or equivalently, the aspect angle to the target changes during the M pulses when the target consists of more than one scatterer. Compensating for the phase and amplitude distortions caused by the slight change in radial distance to each scatterer require known relative positions and strengths of each point scatterer as well as the angular velocity of the target. Fig. 12 shows a scenario with the same type of target as in Fig. 11. The compressed range profile is given as a function of the rotation of the target during the M pulses. Apart from a widening of the target in synthetic range the non-coherent estimate shows little distortion when compared to the coherent estimate. Again, the DNN has not been trained on this type of scenario.

IV. Conclusion

In this study we have investigated the use of a DNN as a non-linear mapping from magnitude or modulus in discrete frequency domain to a corresponding magnitude in time domain. By so doing we have obtained an approximation of a high resolution range profile of targets with a non-coherent radar.

We found that for SNRs $>$ 10 dB the non-coherent DNNs ability to resolve targets within a pulse length was a factor two worse than the equivalent coherent system, i.e. target spacing

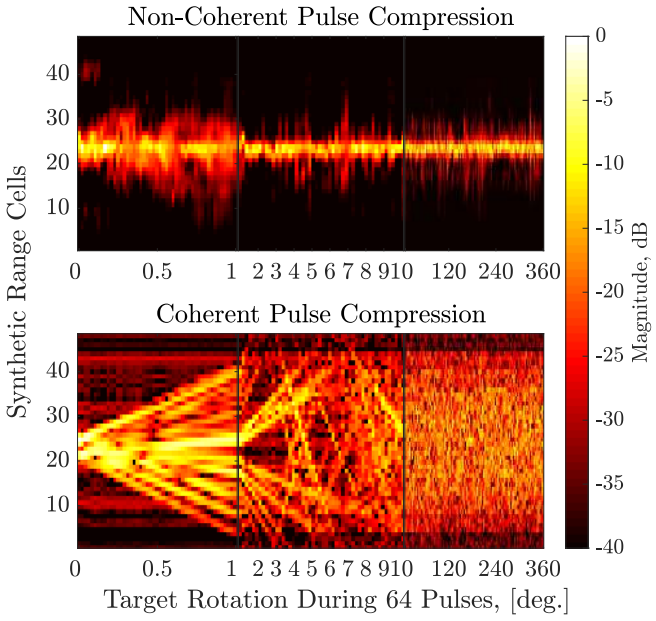

Fig. 12. Scenario with a rotating target. The rotation in degrees during the M pulses is specified along the x-axis for three different intervals. The y-axis shows the $N' = 48$ synthetic range cells. As an example, a rotation of 1° with $M = 64$ pulses and 1kHz PRF corresponds to an angular velocity of 16° per second or 2.6 rpm. The mean SNR was 10 dB.

needed to be increased by factor two before targets peak-to-valley-ratio grew beyond 3 dB. The average peak-to-valley ratio with the DNN solution reached the corresponding value of the coherent peak-to-valley ratio after a target separation of 10 synthetic range cells or 1/3 of the full pulse length.

The gain in SNR was estimated to be greater than 6 dB with 95% empirical probability for SNRs greater than 5 dB. Depending on the target it may be better in terms of detection probability and false alarm trade-off to integrate the required 64 pulses and perform detection on these rather than the DNN output. Provided one can buffer the 64 pulses, an appealing solution is then to use pulse integration for detection of targets and later only apply the DNN solution on the batches with detected targets for increasing the resolution, as this may decrease the processing time compared to applying the DNN solution to all batches.

We have only considered stationary scenarios and a stepped frequency waveform with a fixed interval between frequencies. The extension to dynamic scenarios and random frequency sampling is left for future work. We have however shown examples where the DNN yields reasonable results in dynamic scenarios despite not having been trained on these. This indicates that such DNN solutions could be of benefit even with coherent systems as these are sensitive to phase changes which are unavoidable in dynamic scenarios. We have also not considered the effects of multipath propagation, interference and other forms of noise which is left for future work.

All results in this study are based on simulated data and evaluation on real data is left for future work. The required number of data samples for avoiding overfitting the network was found to be quite high for the chosen architecture, requiring an order of 10^5 samples. Obtaining this amount of real data for training is not a promising solution. We therefore believe that models of targets, such as the example presented in this study, will always be relevant for DNN based pulse compression, at least in transfer learning where the network is first trained on simulated data and then fine tuned on real data. We make no claim to have found the optimal architecture or training procedure for the DNN. Instead we have settled with showing that a rather simple DNN architecture can be used to gain decent results for non-coherent pulse compression. Solving this type of problem with more elaborate network architectures such as the one presented in [15] is left for future work.

The results presented in this study offers new abilities to non-coherently process frequency agile radar signals for both imaging applications and detection. In case of detection one can either add traditional methods such as a constant false alarm rate (CFAR) detector to the presented method, or look at an end to end approach where the network instead outputs a list of all targets with desired attributes using the same type of input. A study on which of these two approaches is better given certain criteria is left for future work.

REFERENCES

[1] Urkowitz, H., Hauer, C. A., and Koval, J. F., "Generalized resolution in radar systems," *Proceedings of the IRE*, vol. 50, no. 10, pp. 2093–2105, Oct 1962.

[2] Levanon, N., "Noncoherent pulse compression," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 2, pp. 756–765, April 2006.

[3] Peer, U. and Levanon, N., "Compression waveforms for non-coherent radar," in *2007 IEEE Radar Conference*, 2007, pp. 104–109.

[4] Seleym, A., "A new non-coherent pulse compression based on binary codes for on-off keying (ncpc-bc-ook)," in *2013 IEEE 20th International Conference on Electronics, Circuits, and Systems (ICECS)*, 2013, pp. 731–734.

[5] Levanon, N., Cohen, I., Arbel, N., and Zadok, A., "Non-coherent pulse compression â aperiodic and periodic waveforms," *IET Radar, Sonar Navigation*, vol. 10, no. 1, pp. 216–224, 2016.

[6] Kayani, J. K. and Hashmi, A. J., "Comparative study of non-coherent pulse compression and non-coherent pulse integration techniques for radars," in *2013 14th International Radar Symposium (IRS)*, vol. 2, 2013, pp. 696–701.

[7] Barton, D., *Radar System Analysis*, ser. Artech House Radar Library. Artech House, 1979.

[8] Seyfried, D. and Schoebel, J., "Stepped-frequency radar signal processing," *Journal of Applied Geophysics*, vol. 112, 11 2014.

[9] Grohs, P., Koppensteiner, S., and Rathmair, M., "Phase retrieval: Uniqueness and stability," 2019, arXiv: 1901.07911 [math.FA].

[10] Fienup, J. R., "Phase retrieval algorithms: a comparison," *Appl. Opt.*, vol. 21, no. 15, pp. 2758–2769, Aug 1982.

[11] Beinert, R. and Plonka, G., "Sparse phase retrieval of one-dimensional signals by prony's method," *Frontiers in Applied Mathematics and Statistics*, vol. 3, p. 5, 2017.

[12] Nishizaki, Y., Horisaki, R., Kitaguchi, K., Saito, M., and Tanida, J., "Analysis of non-iterative phase retrieval based on machine learning," *Optical Review*, 01 2020.

[13] Rivenson, Y., Zhang, Y., Günaydın, H., Teng, D., and Ozcan, A., "Phase recovery and holographic image reconstruction using deep learning in neural networks," *Light: Science Applications, Nature*, vol. 7, no. 2, Feb 2018.

[14] Metzler, C., Schniter, P., Veeraraghavan, A., and Baraniuk, R., "prDeep: Robust phase retrieval with a flexible deep network," ser. Proceedings of Machine Learning Research, Dy, J. and Krause, A., Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 3501–3510.

[15] Masuyama, Y., Yatabe, K., Koizumi, Y., Oikawa, Y., and Harada, N., "Deep Griffin-Lim iteration," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 61–65.

[16] Kerr, D. E., et al., *Propagation of Short Radio Waves*. New York, Dover Publications Inc., 1951.

[17] Doo, S. H., Smith, G., and Baker, C., "Point scatterer modeling of complex targets," in *2014 International Radar Conference*, Oct 2014, pp. 1–6.

[18] Swerling, P., "Probability of detection for fluctuating targets," *IRE Transactions on Information Theory*, vol. 6, no. 2, pp. 269–308, April 1960.

[19] Griffin, D. and Jae Lim, "Signal estimation from modified short-time fourier transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 32, no. 2, pp. 236–243, 1984.

[20] Bendory, T., Beinert, R., and Eldar, Y., "Fourier phase retrieval: Uniqueness and algorithms," in *Compressed Sensing and its Applications: Second International MATHEON Conference 2015*, Boche, H., Caire, G., Calderbank, R., März, M., Kutyniok, G., and Mathar, R., Eds. Springer International Publishing, 2017, pp. 55–91.

[21] Kogan, D., Eldar, Y. C., and Oron, D., "On the 2d phase retrieval problem," *IEEE Transactions on Signal Processing*, vol. 65, no. 4, pp. 1058–1067, Feb 2017.

[22] Balan, R., Casazza, P., and Edidin, D., "On signal reconstruction without phase," *Applied and Computational Harmonic Analysis*, vol. 20, pp. 345–356, 05 2006.

[23] Kingma, D. and Ba, J., "Adam: A method for stochastic optimization," *International Conference on Learning Representations*, 12 2014.

[24] Chollet, F. *et al.*, "Keras," https://keras.io, 2015.

[25] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[26] He, K., Zhang, X., Ren, S., and Sun, J., "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, vol. abs/1502.01852, 2015.

[27] Kayani, J. K., "Performance analysis of noncoherent pulse compression technique," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 4, pp. 2986–2990, 2011.

[28] Taylor, T. T., "Design of line-source antennas for narrow beamwidth and low side lobes," *Transactions of the IRE Professional Group on Antennas and Propagation*, vol. 3, no. 1, pp. 16–28, 1955.

[29] Kayani, J. and Hashmi, A., "A novel non-coherent radar pulse compression technique based on periodic m-sequences," *Aerospace Science and Technology*, vol. 53, 03 2016.

[30] Ma, Y.-B., "Velocity compensation in stepped frequency radar," Master's thesis, Naval Postgraduate School, Monterey California, 1995.

[31] Liu, Y., Meng, H., Hao, Z., and Xiqin, W., "Motion compensation of moving targets for high range resolution stepped-frequency radar," *Sensors*, vol. 8, 05 2008.

Alexander Karlsson received the M.Sc. degree (Hons. Best Graduate of the Year) in electrical engineering from the Royal Institute of Technology (KTH), Stockholm, Sweden, in 2016. He is currently with the Electronic Warfare Division at the defence and security company Saab, Stockholm, where he is pursuing the Ph.D. degree with the Department of Information Science and Engineering at KTH, and the Wallenberg AI, Autonomous Systems and Software Program (WASP).

Magnus Jansson (S'93, M'98, SM'15) received the Master of Science, Technical Licentiate, and Ph.D. degrees in electrical engineering (automatic control) from KTH Royal Institute of Technology, Stockholm, Sweden, in 1992, 1995 and 1997, respectively. During 1997-98 he held a lecturer position with the Control Department, KTH Royal Institute of Technology. Since 1998, he has held various positions in the Signal Processing Department, KTH Royal Institute of Technology: 1998-2003 Assistant Professor, 2003-2012 Associate Professor, and since 2013 Professor. In January 2002 he was appointed Docent in Signal Processing at KTH Royal Institute of Technology. Dr Jansson spent one year, during 1998-99, as Post Doc with the Department of Electrical and Computer Engineering, University of Minnesota. His research interests include statistical signal processing; machine learning, navigation and positioning, sensor array processing, time series analysis, and system identification. He was an Associate Editor, during 2008â2012, and a Senior Area Editor, during 2012â2014, for the IEEE SIGNAL PROCESSING LETTERS. He is an Associate Editor for the EURASIP Journal on Advances in Signal Processing (since 2007) and Elsevier's Signal Processing (since 2015)

Henrik Holter received the M.Sc. degree (Hons. Best Graduate of the Year) in electrical engineering and the Ph.D. degree in electromagnetic theory from the Royal Institute of Technology, Stockholm, Sweden, in 1996 and 2000, respectively. He is currently with the Electronic Warfare Division at the defence and security company Saab, Stockholm, where he is the Head of Research and Technologies. He has authored or co-authored around 50 journal and conference publications. Dr. Holter was a recipient of the IEEE Antennas and Propagation Society 2003 R. W. P. King Award.