In the present reporting period **2022** we implemented the "**random phase**" data generation method. This step was necessary for three reasons: (1) when the training traces were obtained based on the experimental parameters, for all varying parameters the overall datasets can grow very large even for a relatively small number of values for the individual parameters; (2) there was a need for much more variate pulse shapes and FROG traces for the training process; (3) the experimental FROG traces are better resembled with random phases.

1. We employed the **Densenet-BC** model on a **randomly** generated dataset of **40 000 samples**, which was split into training, validation and test sets [80%, 10%, 10%]. Here we focused on the recovery of the **phase**, since this process proved to be the most difficult part. Early running tests showed an unsatisfactory generalization or **overfitting behavior** of the model, therefore we proceeded to fine-tuning the different parameters of the model, while the overfitting behavior was tackled by several regularization techniques.
One important hyper parameter of deep learning models is the **learning rate**, which sets how fast the model will learn, it sets the step of the stochastic gradient descent algorithm. We experimented with different learning rates, and settled for a variable learning rate, where we automatically change its value during a single training epoch. We also tested the model for different **batch sizes** of the input data, for different **growth rate**, **depth** (number of layers) and **compression rate** of the Densenet model. Also, different **weight and bias initialization** methods were tested for the convolution and other layers of the network. The **error** between the predicted and original phases was calculated for **different loss functions** like L1Loss, MSE or custom functions.
Overfitting refers to the over-learning of the training set, when the details and noise in the training data are learned to the extent that the model reproduces almost perfectly the learning data, but does not generalize well, and reconstruction fails for new data. Overfitting may occur due to the oversized capacity of a model. There are several techniques to handle overfitting, here we mention only a few of them. **Dropout** is a technique that randomly sets neurons in a layer to zero according to some probability, therefore reducing the capacity of the model. Another technique is called **weight-decay**, which adds a small penalty to the loss in order to avoid overfitting. Weight decay helps to keep the weights of the model as small as possible, avoiding their growth out of control. Beside these methods we also implemented and tested **early stopping**. We also tried the most straightforward method to tackle overfitting, by reducing the capacity of the model using a fewer number of layers and neurons. These testing processes led to a model, which provided a testing set error (L1Loss) of 1.32 in case of the phase reconstruction. In Fig. 1 we show an example of a FROG trace and the predicted and generated phases obtained by propagating the FROG image through our deep network. The agreement is good, especially in the region, where the amplitude shows significant values, which is generally true for a large percentage of the test set.

2. **Ambiguity removal** proved to be essential for both data generation methods (parametric and random). While earlier we obtained relatively good results in the case of the random model for a dataset of 40000 samples, the parametric model still showed an overfitting behavior. This issue was solved by removing the time reversal ambiguity, which means that both E(t) and E(-t) may lead to the same FROG trace.
3. When both data generation models seemed to be ambiguity free at the time of reporting, we generated larger datasets in both cases and trained separately the neural network for these sets in order to decrease the error on the test set even further and consequently to obtain a better generalizing network. We generated around **120 000 samples** with each model, split into training, validation and test subsets [80%, 10%, 10%]. The training time increases drastically with the size of the inserted FROG images, so the final **image size** was chosen to be 128x128

pixels, which is the size recquested by the most common classical recovery algorithms too. Each FROG image is preprocessed before entering the convolutional neural network such that it has zero mean and a standard deviation of 1 (std = 1). The neural network was trained for several hundred epochs, using a variant of the stochastic gradient descent algorithm, named Adam. Ideally, these optimization algorithms set the many parameters of the network in such a way that the training error between the recovered and original spectral electric fields is minimized.

4. As a next step we inserted **white Gaussian noise** into the FROG images with zero mean and a std = 0.1. After the training of the neural network is completed we test the recovery performance on the test set [~12000 samples]. In both cases we calculate an average test error for three cases: without noise on the images, with a noise equal to the noise level used during training (std = 0.1) and with a noise level double than that used for training. The obtained error values are summarized in the table below:

Table 1: Average reconstruction errors for the parametric and random model and for different noise levels inserted into the FROG traces.

Reconstruction error	**No noise**	**Noise (std = 0.1)**	**Noise (std = 0.2)**
L1Loss (Parametric data)	0.078	0.07732	0.15421
L1Loss (Random data)	0.03274	0.03175	0.04491

The table shows that the highest error is obtained for the highest noise level for both models and that the neural network performs almost equally well for the cases with no noise and for a noise level as high as the level used during training. In Fig. 2 we show a few reconstructed spectral amplitudes and phases, first for the parametric model. Even for various phases, Fig. 2 shows a quite good reconstruction where the spectral amplitude is meaningful. Fig. 3 presents a FROG trace with various noise levels on it and the corresponding reconstructed spectral amplitude and phase. Although for high noise levels the reconstruction of the spectral phase and amplitude is not as good as for lower noise levels, the agreement in the region where the amplitude has significant values is quite good. This is indeed signaled also by the higher reconstruction error in this case, which can be seen along the plotted reconstructed quantities.
A similar behavior of the reconstructed spectral amplitude and phase is observed for our second data generation model, the random phase model. Fig. 4 shows several samples with no noise on the FROG traces and the corresponding reconstructed and original spectral quantities. One can see a good agreement between the reconstructed and original spectral quantities in the domain of significant aplitude values. We plotted samples with reconstruction errors above and below the average test error. Fig. 5 shows the results when different noise levels are inserted on a FROG trace. At low noise levels the reconstruction is quite good, but for the highest noise level we see a poorer reconstruction, confirmed also by the higher error value in this case.
We also measured the **reconstruction time** for a single FROG sample, which is on a **millisecond scale**.

5. These results show that our model is robust against noisy FROG traces, which is encouraging from the perspective of reconstructing experimental FROG traces. On Fig 6 we applied the neural network trained with simulated samples on an **experimental FROG trace**, obtained from the partner at the ELI-NP. We found that the random model performed better for this particular FROG trace. As a reference, we also included in Fig. 6 the reconstructed spectral amplitude and phase from the commercial software, used at ELI-NP (Femto Easy), based on classical, reconstruction algorithms. However, for a better reconstruction, we need to simulate samples with a distribution very similar to the experimental one. A promising area for solving this issue is the use of Generative Adversarial Networks (GANs), which are deep

convolutional network with the ability to sintesize such datasets. Transfer learning may also be used to train our network in order to reconstruct experimental data, but in this case we need enough experimental samples.

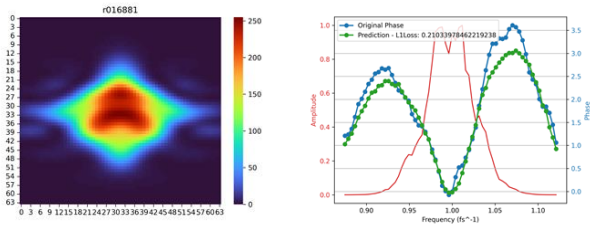

Fig. 1. Left: FROG image. Right: the green line is the predicted phase, while the blue line is the original phase. The original amplitude is shown here just to assess the region, where the pulse has a significant electric field.

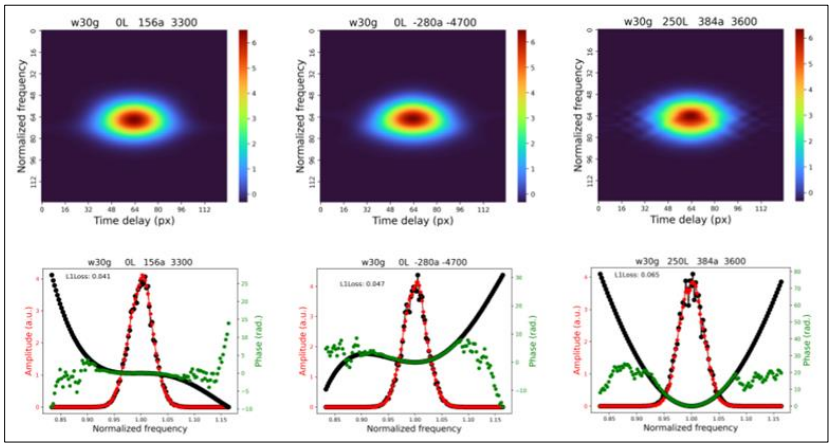

Fig. 2. Parametric model: Black line with circles: original spectral amplitude; Red dashed line with circles: reconstructed spectral amplitude; Black circles: original spectral phase; Green circles: reconstructed spectral phase.

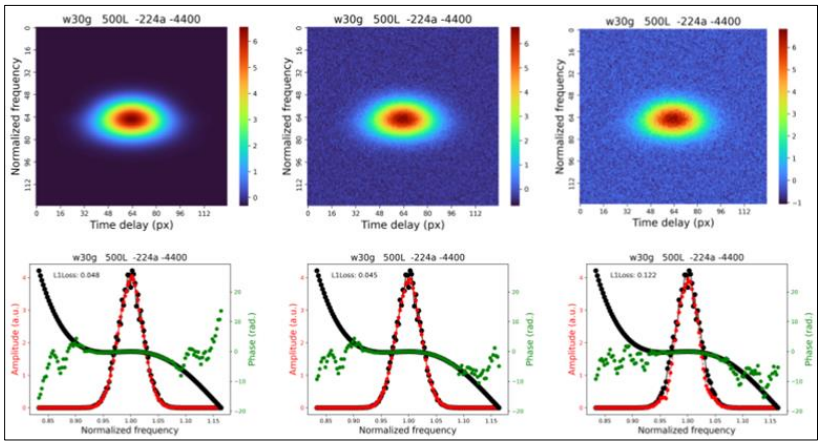

Fig. 3. Parametric model (with noise): Black line with circles: original spectral amplitude; Red dashed line with circles: reconstructed spectral amplitude; Black circles: original spectral phase; Green circles: reconstructed spectral phase.

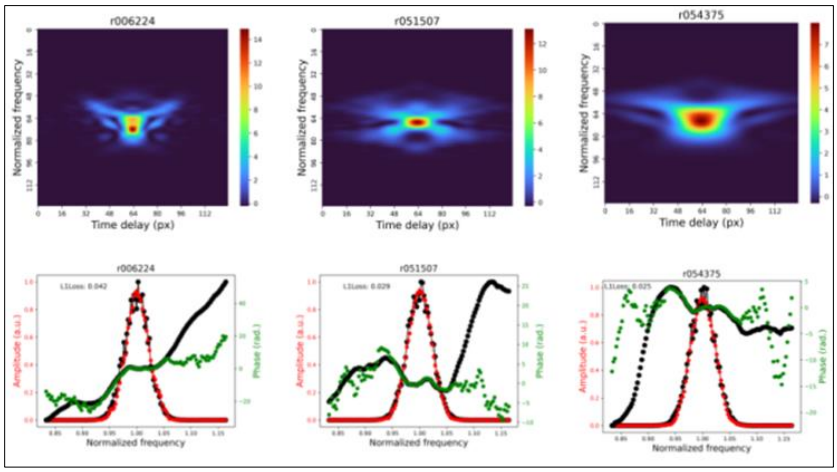

Fig. 4. Random model: Line and circle colors the same as for the parametric model.

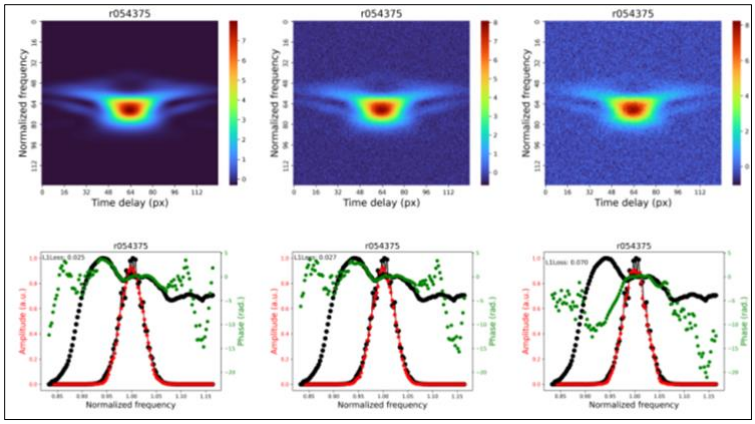

Fig. 5. Random model (with noise): Line and circle colors the same as for the parametric model.

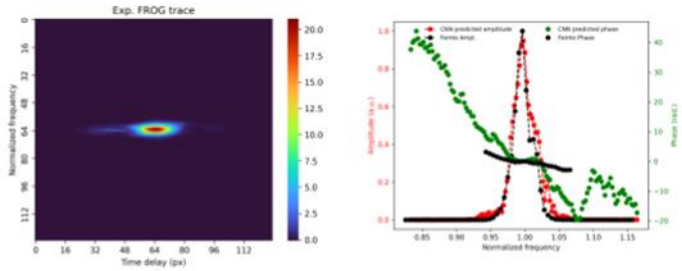

Fig. 6. Experimental FROG trace and the CNN reconstructed spectral phase (green) and amplitude (red). Black circles and lines show the reconstructed spectral quantities by the Femto Easy commercial software.