Norwegian University of Science and Technology
Department of Civil and Environmental Engineering

Machine Learning Methods for Bathymetry Generation in Rivers

Master Thesis prepared by:

Raffa Ahmed Osman Ahmed

Supervised by:

Prof. Knut Alfredsen
Eng. Elhadi Mohsen
Eng. Mahmoud Awadallah

Date: 28th June 2022

Abstract

Natural flood hazards resulted in more than 500,000 reported deaths during the last 20 years. This number is expected to increase in the future due to population growth, rapid urbanization and increased rainfall intensities induced by climate change. Hydrodynamic models are essential tools for simulating flooding events for proper risk management. Topographical and bathymetrical data are essential inputs for hydrodynamic models. One of the promising technology for Topographical and bathymetrical data collection is the LiDAR (Light Detection And Ranging) which has emerged in the last years. Two LiDAR technologies exist; Red LiDAR and Green LiDAR. The former cannot penetrate water surfaces while the latter can. In Norway, Green LiDAR data are collected for only few rivers while Red LiDAR data are available for most of the rivers. This study sough to investigate the suitability of two Machine learning (ML) methods, Artificial neural network (ANN) and Convolutional neural network (CNN), for extracting river bathymetry for Red Lidar data. ML models were trained in Gaula river in Norway with Green LiDAR data. Four different model training scenarios and two activation functions of ANN were tested. The accuracy of the trained ML models was tested at different reaches of Gaula from the ones used for model training and at two other rivers in Norway, Driva and Surna. Moreover, the study investigated the accuracy of the hydraulic simulations by using bathymetrical data generated by the trained ML models. The preliminarily results of the CNN were promising. However, the method was found to be computationally costly and time consuming. For ANN, the selection of input data and the choice of activation functions were found to influence the accuracy of the results. Trained ANN models were able to predict bathymetrical data with reasonable accuracy in most of the cross-sections of Gaula river. Trained ANN could also yield accurate estimation of bathymetrical data when evaluated at Driva and Surna rivers. Prediction accuracy was generally low for cross-sections that vary significantly from the ones used for model training and for cross-sections that contain islands. Regarding hydraulic simulations, terrains estimated by ANN decreased the inundation area error significantly when compared with Red LiDAR simulations. Therefore, well-trained ANN can be used to predict bathymetrical for Red LiDAR in order to improve the accuracy of hydraulic simulations.

Dedication

This master thesis is dedicated

To my beloved and inspiring father may god have mercy on him, who supported me and wished to see this moment,

To my mother, who constantly encouraged me and filled me with love,

To my siblings Ruaa, Rowida, and Fyad my backbone,

To Waleed Babiker,

To Ali Ahmed who made this journey possible with his support, and always being there when I needed him,

To my friends in Sudan and Norway,

To my teachers,

Raffa Ahmed

28th June 2022

Acknowledgement

After two years of studying at NTNU,

I am very grateful and thankful to my supervisor Professor Knut Alfredsen, for his guidance, advice, and his support to finish this master thesis.

I would like to thank Elhadi Mohsen, Mahmoud Awadallah my co-supervisors for their endless support to guide me through this journey and for being always welcome to help.

I would like to thank all the professors at the Hydropower Development Master Program, NTNU for their dedicated and hard efforts, and the time they spent helping us gaining knowledge, I truly appreciate it.

I would like to thank Hiba Abdallah and Duaa Bushara for always being there.

Special thanks to the Sudanese community in Norway.

I would like to thank the Norwegian Agency for Development Cooperation NORAD for the full sponsorship. It wouldn't be possible without their support.

I would like to thank the Norwegian University of Science and Technology (NTNU), Department of Civil and Environmental Engineering student affairs office, and the advisors for their continuous communication throughout the COVID-19 pandemic, while we were studying from our home countries.

Table of Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
DEM	Digital Elevation Model
MSE	Mean Squared Error
Qm	Mean Flood Discharge
Q10	10 years Flood Discharge
Q100	100 years Flood Discharge
Q500	500 years Flood Discharge
ReLU	Rectified Linear Activation Unit
Sigmoid	Soft Step Activation Function

Chapter 1 Introduction

Rivers are crucial natural features that serves a variety of benefits, such as transportation, power production, drinking water supply, ecological system balance and irrigation (Sundt, Alfredsen et al. 2021). On the other hand, rivers can be an initiation point of natural disaster, i.e. flood. The flood damages in Europe is constantly increasing over the time, and minimizing this risk is one of the priorities of the continent. The settlements on the flood plain zones, in addition to the rapid climatic variability and environmental changes due to technology enhancement, such as land use urbanization, are significantly affecting the flood risk (Kron, Eichner et al. 2019). Moreover, the global warming and snow melt led to earlier soil saturation and increased runoff (Bloschl, Hall et al. 2017). In addition, river floods vary in spatial and temporal scale (Shrestha and Nestmann 2009). All these factors need regular studies about the floods impact and the following consequences as well as local planning.

Hydrodynamic models are extremely substantial tools that help the practitioners studying and analysing the impact of flood for risk assessment and natural hazard management. The input data for these models vary depending on the used tool and selected purpose of modelling. Each tool has a fundamental concept based on selected field of science, such as hydrology and hydraulics (Shrestha and Nestmann 2009). One of the most commonly used hydrodynamic models is perhaps the Hydrologic Engineering Center River Analysis System model (HEC-RAS) by the United States Army Corps of Engineers (Alaghmand, Abdullah et al. 2012). In 2D flood simulation using HEC-RAS, the main input data are runoff hydrograph for the selected river, upstream and downstream boundary conditions beside the topographical data and defined propertied for the landuse. The topographical data is a foundational input to the model that describes the geometries of the riverbed and the flood plains.

Several topographical and bathymetrical data collection methods exists for academic and practical applications. Riverbed data collection can be manually done using shipboard and ecosounder (Gao 2009), or Acoustic Doppelr Currnet Profiler (ADCP) by maneuvering through the river. This approach generates accurate riverbed measurements. However, it is labour-intensive , costly and time-consuming. Remote sensing is another method for collecting topographical and bathymetrical data. It is becoming a major method in data collection techniques and has a wide range of applications because it provides lifelike images (i.e. Google Earth) that serve many fields (Zhang, Zhang et al. 2016). Light Detection And Ranging (LiDAR) is a remote sensing airborne technology that calculates the round distance from the light transmitter sensor to the ground object (Meng, Currit et al. 2010). LiDAR sensors provide fast and highly quality elevation measurements with three dimensional coordinate system, comparison to other surveying systems like photogrammetric system (Meng, Wang et al. 2009). Recently, LiDAR attracted attention in bathymetrical data collection of rivers (Fonstad and Marcus 2010). To measure the riverbed, LiDAR uses the green light sensor which operates in the green part of electromagnetic spectrum which can penetrates the water and recover the reflection from the riverbed (Kinzel, Legleiter et al. 2013). The above-mentioned point made the topographical data generated from LiDAR emerged as one of the most significant input for Hydraulic simulation tools and flood assessment.

As mentioned before, Green LiDAR penetrates the water unlike the Red LiDAR which consider the rivers as flat surfaces and being absorbed by the water column (Kinzel, Legleiter et al. 2013). Many studies carried out to minimize the DEMs errors, manually or by systematic editing. The former studies used explicit algorithms to correct noises across the DEM (Stevenson, Sun et al. 2010). Studies investigated the discrepancy between the bathymetry and topographical data were done, i.e.(Allouis, Bailly et al. 2010).

Recent studies investigated the application of machine learning techniques in flood assessment and natural disasters management. This is due to the fact that fast dynamic of urbanization is involving different scientific disciplines led to large number of data (Deparday, Gevaert et al. 2019). Many studies were done to improve the DEMs using ANN. For instance, (Stevenson, Sun et al. 2010) and (Wendi, Liong et al. 2016). In this research machine learning methods were used based on the characteristic of handling nonlinear data, beside the ability of processing large datasets.

In Norway, Green LiDAR was collected for only few rivers. In contrast, Red LiDAR data is covering wide space among the country and it is readily available for almost all rivers. Hence, this study aims at finding a suitable ML method to obtain the river bathymetry (wetted zone) from the Red Lidar for accurate flood modelling. The specific objectives of the study are:

- Evaluating the suitability of different ML algorithms in predicting river bathymetry for Red Lidar data.
- Evaluating the accuracy of trained ML models in different rivers
- Investigating the accuracy of hydraulic simulations using bathymetrical data generated by trained ML models

Chapter 2 Literature Review

2.1 Topography Data

There are many types of topography data generation sources that vary in their collection methods and accuracy. For instance, ground surveys, digitizing hardcopy topographic maps, photogrammetry, interferometric synthetic aperture radar (IfSAR), and light detection and ranging (LiDAR) (Hodgson, Jensen et al. 2003, Muhadi, Abdullah et al. 2020).

In Situ ground survey can have higher resolution depending on the measurement device's accuracy and the spatial distribution of the measured points. It is suitable for small-scale studies and can be very accurate (Casas, Benito et al. 2006). Traditional leveling equipment and advanced ones like total station could produce elevation maps. However, both are time-demanding and can bare a trade between spatial and temporal accuracy. Therefore, the photogrammetry tools have made it possible to acquire higher density from a small distance and even faster. Two types of photogrammetry are available: aerial (camera on the air like a plane) and terrestrial (mounted on a tripod). In the past mechanical methods were used to extract the elevation information from the photos taken by the cameras. Then with the help of computers, analytical tools were developed to create the topography maps, which allow automation of the process and creation of digital elevation models (DEMs) 100 times faster than the previous method. Lastly, data acquisition has become automatic through digitalised photogrammetry with high spatial resolution, which can reach up to 0.01 to 1 m cell size for small areas (Lane, Richards et al. 1993, Chandler 1999, Lane, James et al. 2000, Bird, Hogan et al. 2010).

Digital photogrammetry often cannot accurately represent the bathymetry of the rivers. Also, it is not possible to work in all weather conditions. In this manner, coupling the latter with total station measurement can lead to efficient results. Although these improvements can save some time, it still takes longer to create a controlled network and can have a limitation concerning visibility between the device and the point. Thus, the use Global Positioning System (GPS) can tackle these limitations and achieve accuracy of (2±3 cm) topographic data (Brasington, Rumsby et al. 2000, Casas, Benito et al. 2006).

IfSAR working principle is based on signals emitted from microwave sensors to the earth's surface, then the scattered signal will be recorded. Two images are combined to generate the DEM using the interferometric SAR. There are two types: spaceborne IfSAR and airborne IfSAR. Airborne IfSAR can be more flexible than the other one and can have higher resolution. One famous DEM created from the spaceborne is the shuttle radar topography mission (STRM). The advantage of using IfSAR is that larger areas can be covered at any time (even at night) and in any weather conditions. The accuracy provided with airborne IfSAR is about 5 m spatial resolution and 0.5 – 3 m vertical resolution. However, in urban areas or densely vegetated areas, IfSAR has some limitations with the complexity of the structures, which scatters the signals or cannot penetrates the canopy surfaces (Dowman 2004, Mercer 2004).

Lastly, the new emerging technology, LiDAR, has the same concept as IfSAR with different sensor types. LiDAR systems are based on using a laser sensor to scan the objects and measure the distance between the platform, which usually is an airplane and the surface of reflection. Like IfSAR, GPS/INS system is installed in the aircraft to link the location information with the collected data. LiDAR data can be collected from the ground, terrestrial LiDAR, helicopter or airplane, airborne LiDAR, or space, spaceborne LiDAR. An example of the latter is Geoscience Laser Altimeter System (GLAS). LiDAR has some advantages compared to the other methods. Such as, data can be collected anytime during the day, even in cloudy conditions, very efficiently. Also, it can have better penetration for the vegetation covers and better representation of urban areas. Kraus and Pfeifer (1998) stated that LiDAR can produce DEM in forest areas, similar to the accuracy of photogrammetry in open areas. The accuracy of the DEM generated by airborne LiDAR is 0.5 – 2 m spatially and 10 cm vertically (Hodgson, Jensen et al. 2003, Dowman 2004, Mercer 2004).

2.2 Application of LiDAR data in flood:

Mapping flood areas is an important task when dealing with risk assessment. A crucial factor for producing inundation maps is DEM. Webster, Forbes et al. (2004) used airborne LiDAR to create high-resolution DEM to assess the sea-level rise and climate change risk in a coastal area. The flood model from the generated DEM accurately mapped the flood zones. Bales, Wagner et al. (2007) prepared 1.5 by 1.5 m DEM from airborne LiDAR with vertical accuracy of 20 cm to make flood inundation area maps for study sites in North Carolina. HEC-RAS model was developed for the study area to simulate water levels. The difference between observed and simulated water levels was beneath 25cm.

Sampson, Fewtrell et al. (2012) used terrestrial LiDAR to develop DEM with a resolution of 10 cm to simulate an urban flood in the UK. Two models were implemented, LISFLOOD-FP and ISIS-FAST. The result showed how small details in the topography, like road pumps, can influence the flood simulation. The LiDAR well-represented these features, and as a finding when finer details are required like in the case of urban flood modeling, terrestrial LiDAR can be used.

Chen, Krajewski et al. (2017) investigated the applicability of using LiDAR data in flood modelling. LiDAR data was collected during a flood and classified as flooded and non-flooded points. The resulted flood DEM has been subtracted from earth DEM. The root mean squared error was 30 cm compared to high water marks. This study demonstrated the potential of using LiDAR data in flood inundation maps and calibrating and validating flood models.

Santillan, Makinano-Santillan et al. (2016) illustrated the application of LiDAR when combined with flood models in determining the vulnerability of buildings. Abdalla, Pons et al. (2021) introduced a method for collecting finer resolution LiDAR data to be used in flood. The method was tested in small mountain area and large urban area to see the time needed for collecting the data. It took 5 h and 2 days respectively. In the end, the model results showed how the additional gained information from developing very high resolution DEM improved the simulation and can be more reliable.

2.3 Neural Network

Neural networks are resembling for human neural cells, it was inspiration from the brain system complexity, it started with one layer of neurons (Montesinos-López, Montesinos et al. 2022), then the shallow neural network SNN consisting from 1 or 2 hidden layer (Feng, Zhou et al. 2018). The artificial neural networks ANNs kept developing to multiple hidden layers resulting in deep learning systems.

Data-driven models are widely used in hydrological applications where sufficient data sets are available. These models build a mathematical relationship between observed and simulated data. Machine Learning is an advanced type of these models.

Neural network (NN) was adopted in many applications in hydrology. To begin with, rainfall-running models where NN models were preferable compared to physical models. Especially when model parameters are difficult to measure and can have uncertainties when calibrated based on assumptions. Furthermore, complex systems can not be fully understood to be represented by physical equations (Machado, Mine et al. 2011, Bloschl, Hall et al. 2017). Also, ANN model has been used as a surrogate for a process-based model to reduce the computational time needed to calibrate or run many scenarios (Semiromi, Omidvar et al. 2018).

In the field of groundwater, ANN has been used for parameter estimation for the groundwater flow model (Shigidi and Garcia 2003), groundwater level prediction (Sreekanth, Geethanjali et al. 2009, Semiromi, Omidvar et al. 2018, Roshni, Jha et al. 2020), groundwater quality (Jinlong and JinTao 2009, Kulisz, Kujawska et al. 2021, Stylianoudaki, Trichakis et al. 2022), and Karst discharges (Cheng, Qiao et al. 2021).

Moreover, NN models were used in processes related to the hydrological cycle. For instance, rainfall forecasting (Khaniani, Motieyan et al. 2021), predicting evaporation (Algretawee and Alshama 2021, Sivastava, Naidu et al. 2022), downscaling temperature and precipitation data (Wang, Huang et al. 2020), and global climate models (Weber, Corotan et al. 2020). Other applications include water level and inflow in reservoir prediction (Qi, Zhou et al. 2019), and land cover maps (Abdi, Samadzadegan et al. 2018).

Chapter 3 Method and Tools

This section presents the two main methods used for predicting the river cross-sections, image inpainting, and 1-D machine learning training. All the data were prepared with the help of ArcGIS Pro. Different scenarios were also examined for the 1-D trained model. The results from the prediction were tested in a flood simulation model.

Machine learning concept was applied for both model trainings. Machine learning means computer algorithms for data analysis to derive intelligent predictions on an iterative process without explicit programming (Das and Behera 2017).

3.1 Image inpainting

Image inpainting is a method to fill the missing data in image pixels by taking advantage of an existing relatively large data set with similar pixel values (Bertalmio, Sapiro et al. 2000). Several methods for image inpainting can fill the missing data. The technique used here is based on a deep learning convolution neural network for image processing CNN. The method has different applications, for example, revert deterioration, add or remove an element, remove scratches from images, etc.

3.1.1 Convolutional neural network CNN

CNN is a deep learning neural network that uses multiple hidden layers for image analysis or classification problems (O'Shea and Nash 2015). The multi-hidden layers are: convolutional layers, non-linearity or ReLU layers, pooling layers, and fully connected layers. Each one will be explained in further detail.

3.1.1.1 Convolutional Layers

If we have image pixels as input data for the CNN model and this image has features to be detected through the deep learning procedure, the filter will work as a feature patterns detector convolving over the entire image with a specific stride. In each convolutional layer, we should specify the number of filters for pattern detections, such as edges detector, face detector, or other different features, as shown in Fig 3.1.

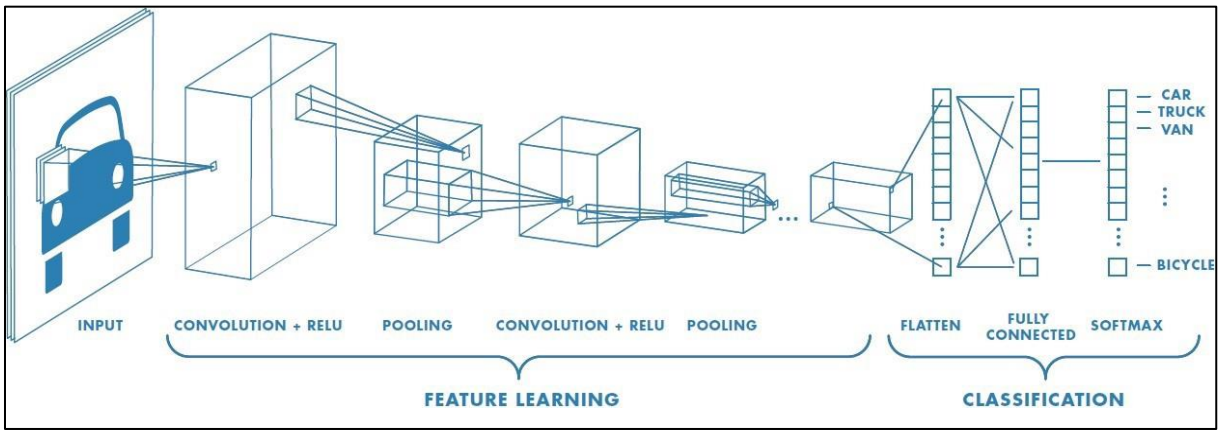


Fig 3.1 Convolutional Neural Network Layers (2018)

3.1.1.2 Filters

A filter can be a relatively small matrix where we decide the number of rows and columns defined also by kernel size (Albawi, Mohammed et al. 2017) . Values within the matrix are initialized with random numbers; for example, if we have a filter 5*5, the filter will convolve over every 5*5 pixels.

We need a filter with a selected dimension on a specific convolutional layer. When this convolutional layer receives input, the filter will slide over each image patch with a size similar to the filter. As shown in Fig 3.2, the filter slides over each 3*3 pixel from the input itself until it moves over each 3*3 pixel of the entire image. The sliding process is referred to as convolving, which multiplies the input by the filter element-wise multiplication and then calculates the summation.

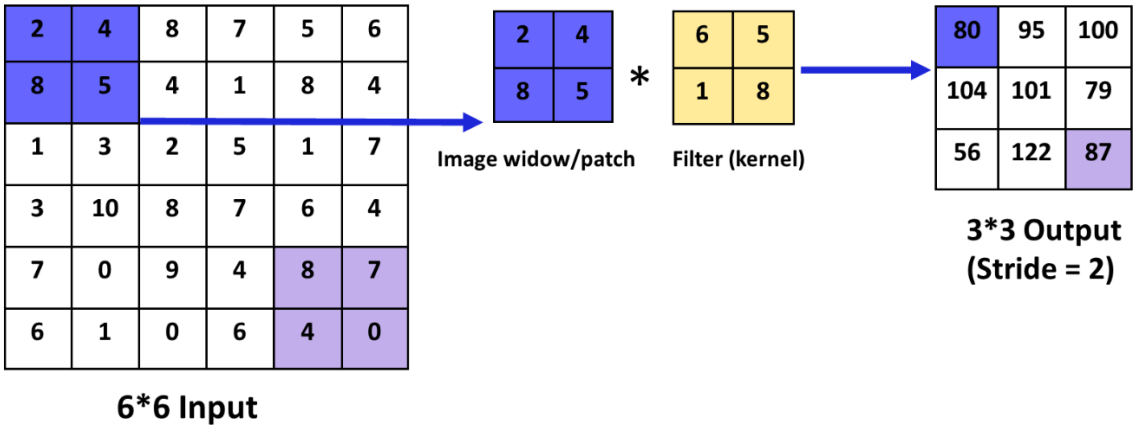


*Fig 3.2 Filter convolving over 6*6 input feature*

3.1.1.3 Non-Linearity Layer

The convolutional layers are typically linear networks when applying the filter for weight and bias calculation. Nevertheless, to apply image processing such as classification, non-linearity is required. This is achieved by using the activation function in most cases. ReLU has recently been the most popular activation function due to the fast convergence and fast processing.

3.1.1.4 Pooling Layers

Pooling is a method used for dimensionality reduction to minimize the computation expensiveness in model training and fasten the major features detection. Pooling has the same mechanism concept as filters, but statistical operations are addressed instead of the element-wise multiplication. It has different types: max pooling, average pooling, global max pooling, and global average pooling. These types apply statistical operations like maximum and average for the pool of values.

For example, in Fig 3.3 we have 4 windows and filter 2*2 with stride 2. The output values are the maximum value in each window; therefore, the dimension will be reduced from 4*4 to 2*2, giving a new image representation.

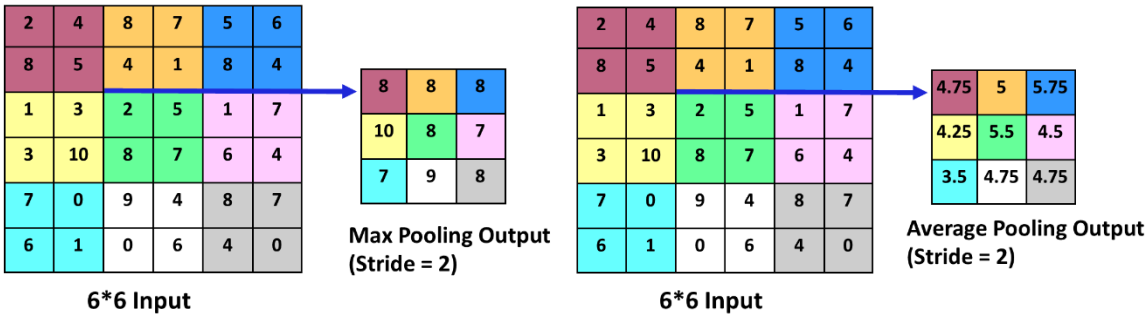


*Fig 3.3 Max and Average pooling illustration example for 6*6 input image*

3.1.1.5 RGB Colour Model

RGB is a red-green-blue colour scheme that defines the image colour. The three colours are added together to reproduce diverse different colours. In CNN image processing, each colour is encoded by numbers from 0 to 255, where 255 is black, and 0 is white (brightest). Colours other than red, green, and blue can be represented by 3 different values "R G B". For example, "210 0 210" is an RGB representation of the magenta colour. Also dark blue is R = 0, G = 0, and B = 100 or "0 0 100"

3.1.2 Image inpainting and CNN

Compared to other methods, this method exploits the avoided post-processing for the images since it uses partial CNN, which robustly works on predicting missed pixel values. The predicted image will no further need blending operations. (Liu, Reda et al. 2018)

Partial CNN means no modification will take place on non-hole regions on the image where the pixels have missing data; instead, the filters will only operate on the unmasked pixels. The process will look into local regions rather than the whole image, efficiently decreasing the number of parameters. In addition, the code has an automated updating mask in each step where the CNN kernel was successfully predicted, resulting in filled pixels. Hence, decrement of the operations in each successive time step, and the filled valid pixels will be part of predicting the missing values in the next time steps (Liu, Reda et al. 2018).

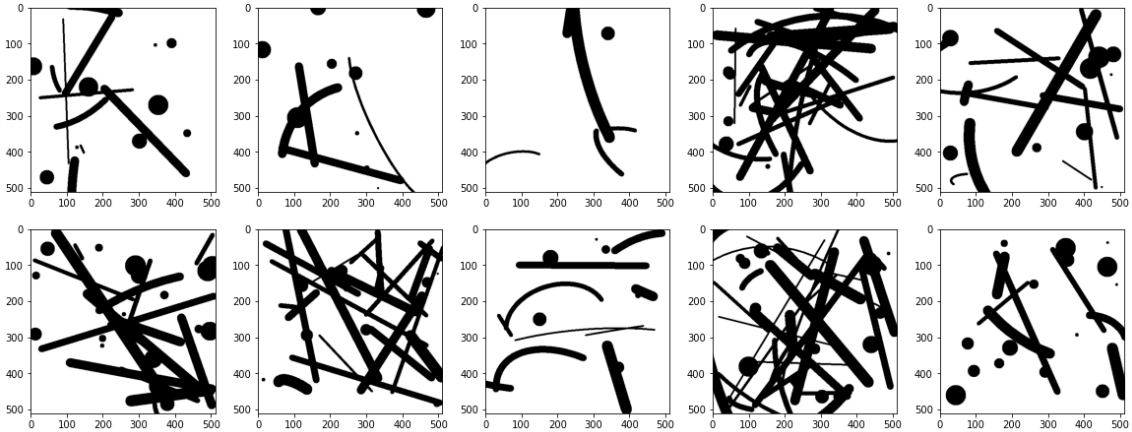


Fig 3.4 Random Generated Binary Mask for CNN Operation (Liu, Reda et al. 2018)

This study sought to exmine the potential of image impating using CNN for predicting river subsurface bathymetry of Red Lidar data. The general idea of this method was to train the CNN model to predict the missing pixel value based on the neighboring RGB pixel values. Green and Red Lidar have the same topographical values and differ only on subsurface bathymetry.

Therefore, the concept was clipping large number of of Green LiDAR images with the masked river as input and a sufficient number of targeted outputs to have a fully trained model with the capability of predicting the missing pixel values on areas with Green Lidar data to evaluate the model performance. Fig 3.5 presents an example of masked image data From Gaula river for CNN training. A trained CNN model can be used for predicting river bathymetry for rivers where only Red LiDAR are available. The predicted bathymetry can be used as an input for hydraulic models to determine inundated areas for vaying flood discharge scenarios

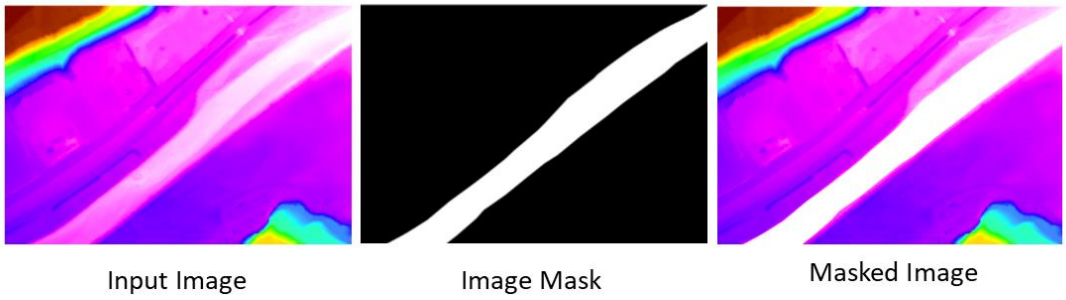


Fig 3.5 An Example of the Input Masked Image from Gaula for CNN Training

3.1.2.1 Preliminary Test for the Method

Data used on a preliminary test for the method was a bathymetry image clipped from a Green LiDAR image with multi RGB values representing the elevations. The image was clipped from Gaula DEM. The tested image was one fixed image as shown on the left image in Fig 3.6, while the mask (middle image) was rotating over large masks database. This implies, the training was done on different shapes of masks. The test used the image inpainting algorithm by (MathiasGruber 2019) as machine learning operator for the CNN model. At the same time, the full clipped image is used as the targeted output for the model.

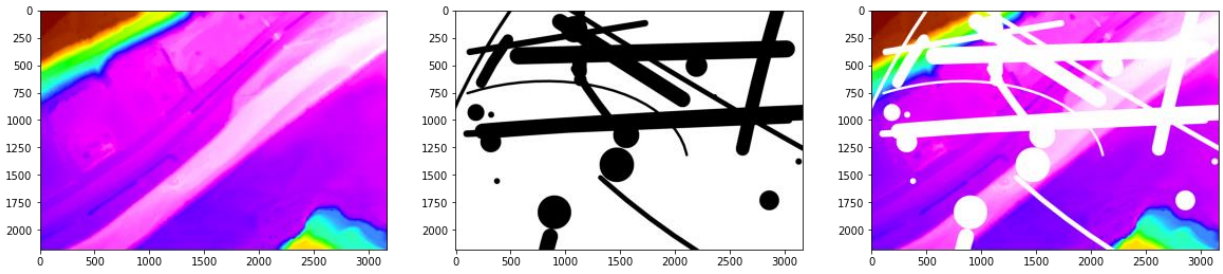


Fig 3.6 An Example of the Randomly Generated Mask on Gaula River

3.2 1-Dimensional training

3.2.1 Artificial Neural Network

Artificial neural networks are computation systems consisting of fully or partially connected neurons or nodes (Yegnanarayana 2009). ANN structure consists of an input, hidden, and output layer. Each layer has a specified number of neurons that transmit the information to other neurons through mathematical equations. The receiver node processes the equation and transfers it to the next layer until the output layer (Fig 3.7).

The mathematical signal applied to the network training is based on weight (w) and bias (b). They are the learnable/trainable parameters that transform the data to find the best fitting model between the input and the output, as represented in f (W_1X_1 + W_2W_2+…W_nX_n +b) Eq *3.1*, where O is a neuron output, X is changeable variables. Each neuron will receive a weighted sum of input from the previous layer plus the bias.

Bias values are the threshold to determine whether the neuron is activated and be passed forward to the rest of the network, considering more model flexibility. Flexibility means more neurons will be activated for forward propagation.

$$O = f\ (W_1X_1 + W_2W_2+\ldots W_nX_n +b) \qquad \text{Eq 3.1}$$

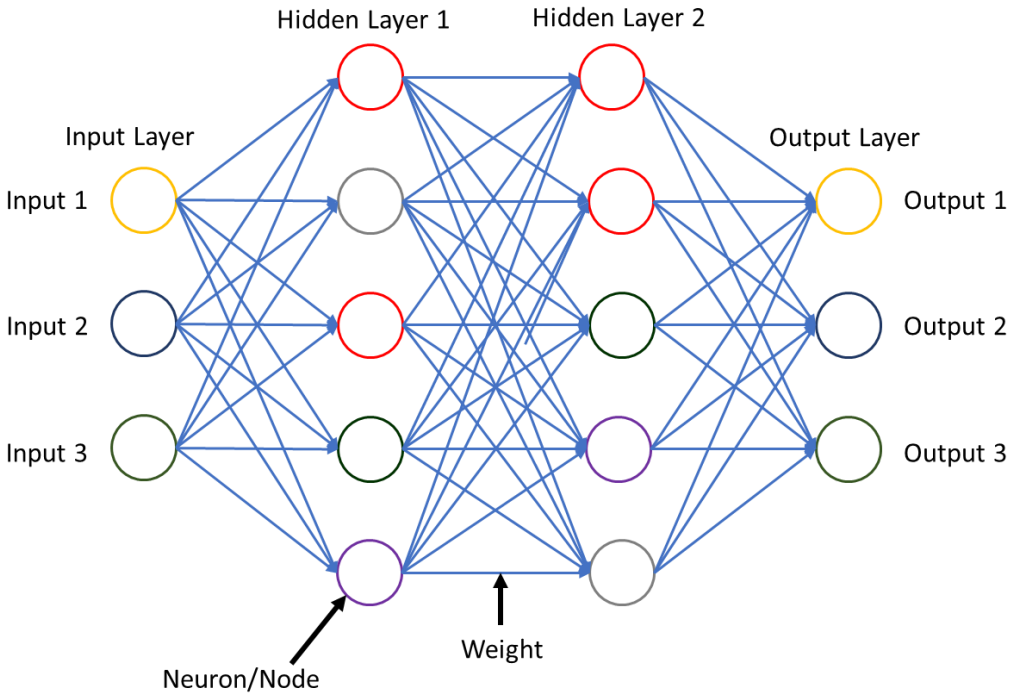


Fig 3.7 General Structure of Artificial Neural Network

3.2.2 Activation function

Machine learning algorithms will learn linear regression fitting straight lines to the training set as a first step. In this case, not all the data will reasonably be represented and capture the true relationship between the input and output data. Therefore, the activation functions are essential tools for adding nonlinear properties to the model output since the model will have considerable bias. In other words, the bias will help the activation function in increasing the model flexibility to fit the data. The weighted sum added to the bias received from a neuron will be passed to the activation function.

The activation function has many types, 3 of them have been tested here (Linear, ReLU, and Sigmoid) as shown in Fig 3.8. The selection of the adequate activation function has many factors, such as the data type and the targeted task.

3.2.3 Loss Function

The loss function or cost function is one of the structured parameters in building a training model in machine learning. It is a method of quantifying the error between the fed or targeted output and the algorithm output. Several loss functions are used in measuring the error. Each of these types has a different algorithm for loss evaluation. In this model, the MSE is used to

find the squared difference between what the model predicted and the targeted output (e), then finding the average as shown below in Eq 3.2.

$$MSE = \frac{e1^2 + e2^2 + \cdots en^2}{n} \qquad \textit{Eq 3.2}$$

The loss calculation is executed at the end of each training step or epoch. It will be updated constantly since the model weight and bias are changing in forward and backward propagation. The loss is calculated for both training and validation.

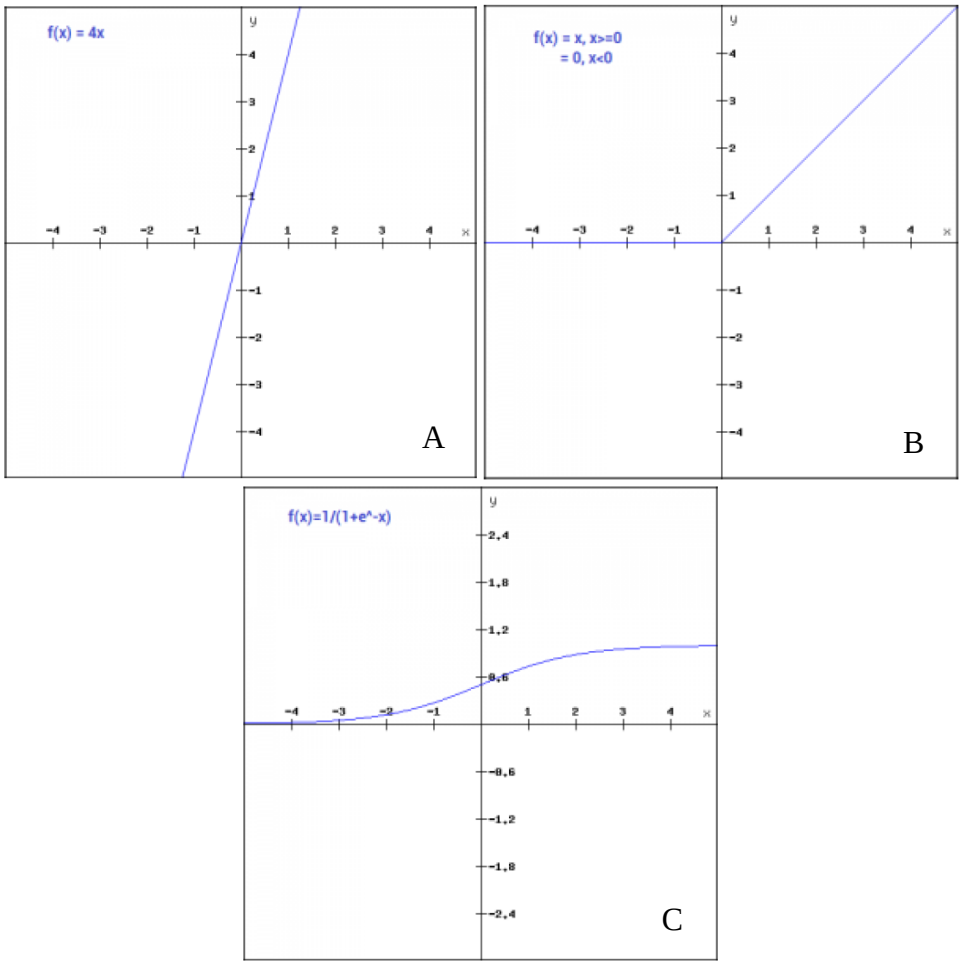


Fig 3.8 A, B, and C are Linear activation function, ReLU activation function, and Sigmoid activation function respectively (Sharma, Sharma et al. 2017)

3.2.4 API Keras

In this method, neural network high-level API code Keras was used for easy implementation of neural networks. It is written in Python and runs on a machine learning platform. Keras is one of the robust machine learning problem solvers. It has a high iteration velocity; it uses deep learning frameworks such as TensorFlow as a backend for faster computation.

In Keras, we can build a sequential model for machine learning problems. The sequential model can simply be built in Python as shown in Fig 3.9. Keras sequential is defined as a "model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor" (TensorFlow 2022).

The full sequential model architecture includes the number of hidden dense layers, input shape, and the suitable activation function connected to each layer.

```
# model creation & adding layers
model = Sequential([
    Dense(164, activation= "relu", input_shape=(400,)),
    Dense(164, activation= "relu"),
    Dense(400, activation= "linear"),
])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 164)               65764

 dense_1 (Dense)             (None, 164)               27060

 dense_2 (Dense)             (None, 400)               66000

=================================================================
Total params: 158,824
Trainable params: 158,824
Non-trainable params: 0
_________________________________________________________________
```

```
# compile model
model.compile(optimizer ="rmsprop",
              loss= "mse")
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 164)               65764

 dense_1 (Dense)             (None, 164)               27060

 dense_2 (Dense)             (None, 400)               66000

=================================================================
Total params: 158,824
Trainable params: 158,824
Non-trainable params: 0
_________________________________________________________________
```

Fig 3.9 Keras Sequential Model Structure and Compilation

To summarize the neural network training workflow, input data is passed into a structured model, the weights are multiplied by each neuron variable and added to bias, and the output is passed to an activation function for non-linearities. This will be a definition for the neurons in

the next layer. This operation is called forward propagation as we propagate the information from the first input layer to the final output. When the output value is calculated, the difference error between the output and the calculated output is found. This loss is used to compute the partial derivative with respect to the weights in each layer propagating backward, and then the weights will be updated. The process is repeated for each epoch until the error is the smallest.

3.2.5 Data

Terrain

The data used in this study has been prepared with the help of ArcGis Pro, using the river terrain for both topography and bathymetry raster images. The Green LiDAR is a representation of the true river bathymetry. The Red LiDAR is considered in this study to have identical topographical data with Green LiDAR, and minor discrepancies are ignored, except for river wetted zones. Gaula, Surna, and Driva are selected for the model training and testing. The LiDAR terrain was generated by the Norwegian Mapping Authority on Høydedata.no. The DEMs has resolution of 0.25 m for Driva and Gaula and 0.5 m for Surna (Awadallah, Juárez et al. 2022).

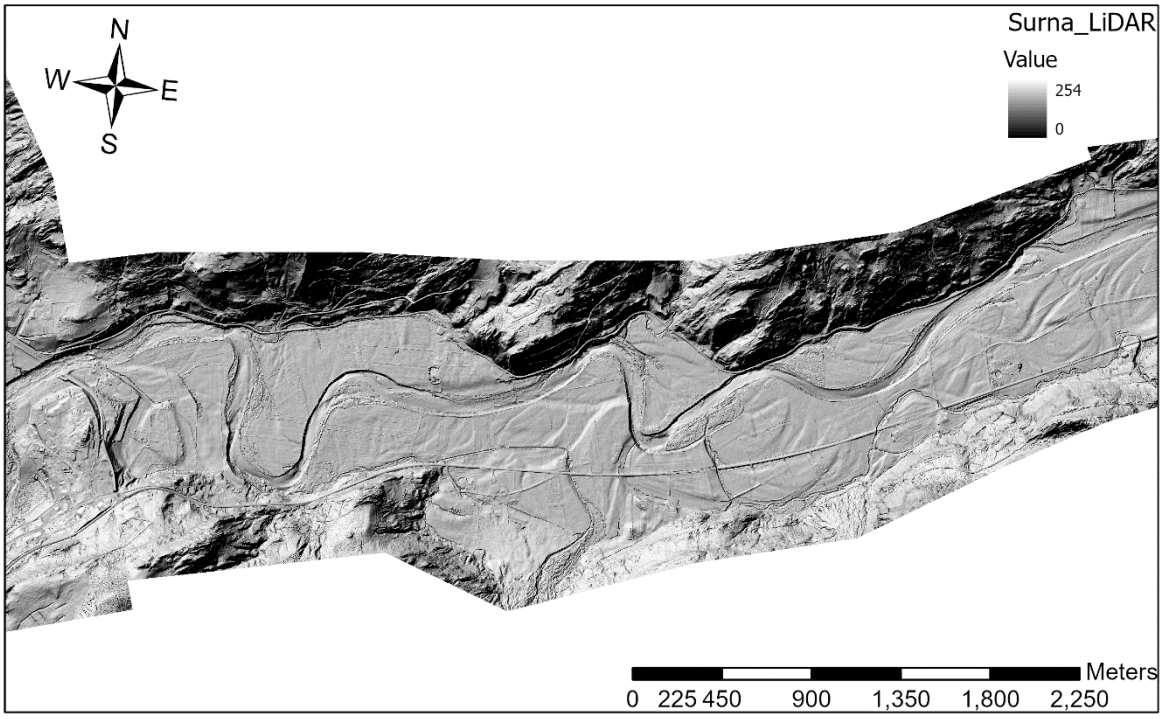


Fig 3.10 Green LiDAR terrain for Surna river

Flood Data

The flood values have been selected from a previous study based on the Norwegian Water Resources Directorate NVE NEVINA.no. The flood scenarios are calculated with regional flood frequency analysis.

Table 3.1 Tested Flood in m^3/s Scenarios Based on NEVINA.no (Awadallah, Juárez et al. 2022)

River	Qm	Q10	Q100	Q500
Driva	545	725	960	1115
Surna	171	230	306	355

Data Preparation Using ArcGIS Pro

ArcGIS Pro supported by ESRI, is the tool used for data preparation. It was used to prepare the input training data for the ANN algorithm model and the hydraulic simulation.

For the input training data, the DEMs for Green and Red LiDAR for Gaula were used for cross sections extraction. The mainstream polygons for Gaula, Driva, and Surna were taken from this study (Awadallah, Juárez et al. 2022). The polygons were edited to fit the mainstream boundaries. A centreline was created for the existing polygon. Considering the observed average river width, 200 m length transects were created each 10 m distance along the river centreline. After that, points were generated along the transects every 0.5 m for elevations values extraction, then the average pixel value from neighboring pixels was extracted into the points for both Green and Red LiDARs. The attribute table now contains the Green and Red LiDAR elevation points value for each transect along Gaula.

The attribute table was extracted as a CSV file to provide input data for the ANN model and easier implementation.

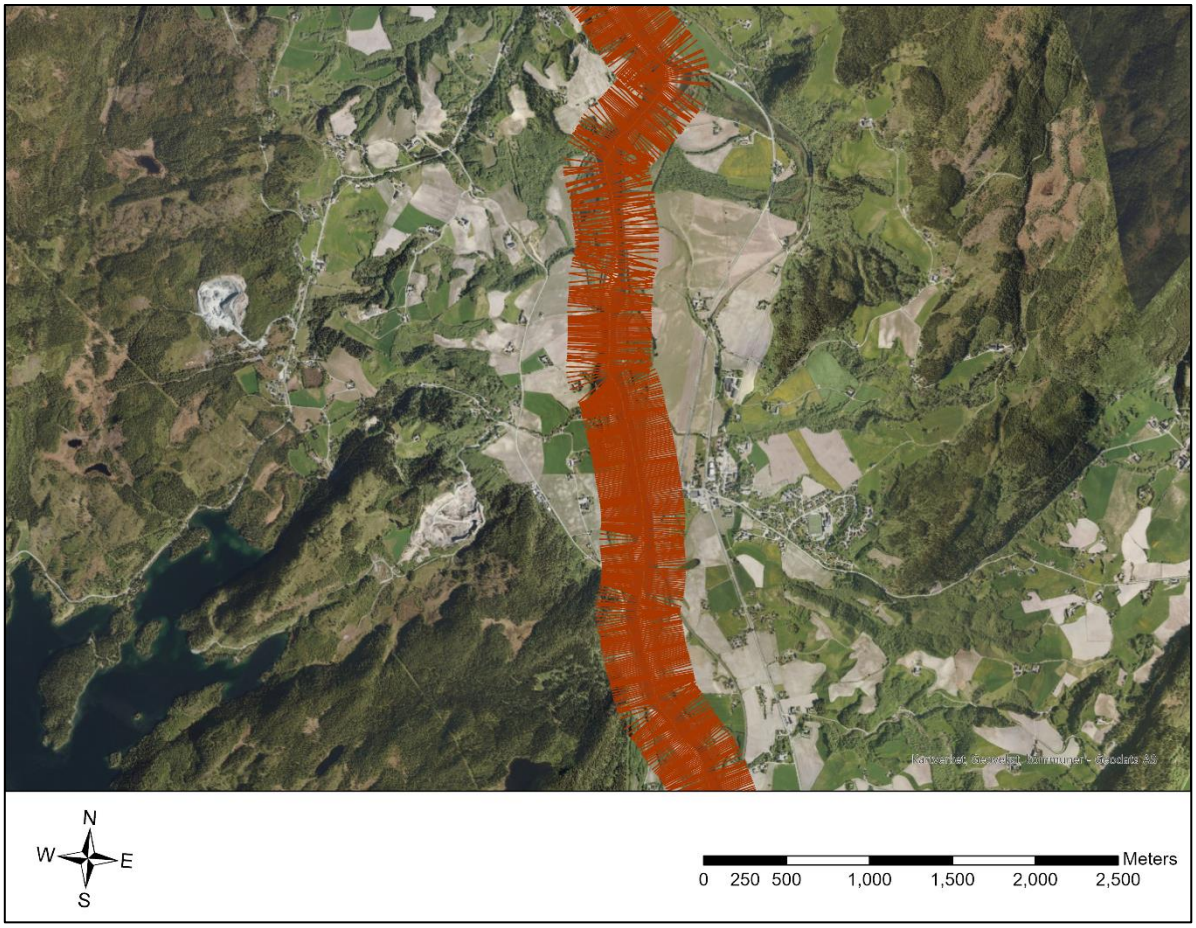


Fig 3.11 Transects and Points Along Gaula River

The predicted bathymetry points were extracted from Python as a CSV file with X, Y coordinates system ETRS1989_UTM_Zone_32N and imported into the ArcGIS program. The predicted cross-sections contained a prediction for the topography elevation values (riverbanks) that were covered by Red LiDAR and not targeted in this study. Therefore, the predicted points were clipped to keep only the wetted zone using the geoprocessing clipping tool. The clipped points transformed into raster as shown in Fig 3.12 B, the raster's cell size is 0.25 m for Driva and 0.5 m for Surna. To test the flood inundation coverage, the Green LiDAR terrain was

clipped using a generated polygon domain, then the mainstream was erased, this will give the geometry for the banks as illustrated in Fig 3.12 A. The predicted bathymetry raster and the Green LiDAR topography domain raster merged into one terrain and processed to fill missing pixel values (Fig 3.12 C) to achieve a DEM with identical banks for the accurate bathymetry and prediction. In addition, the residual between the Green LiDAR and predicted bathymetry was found. This procedure has been done for Driva and Surna.

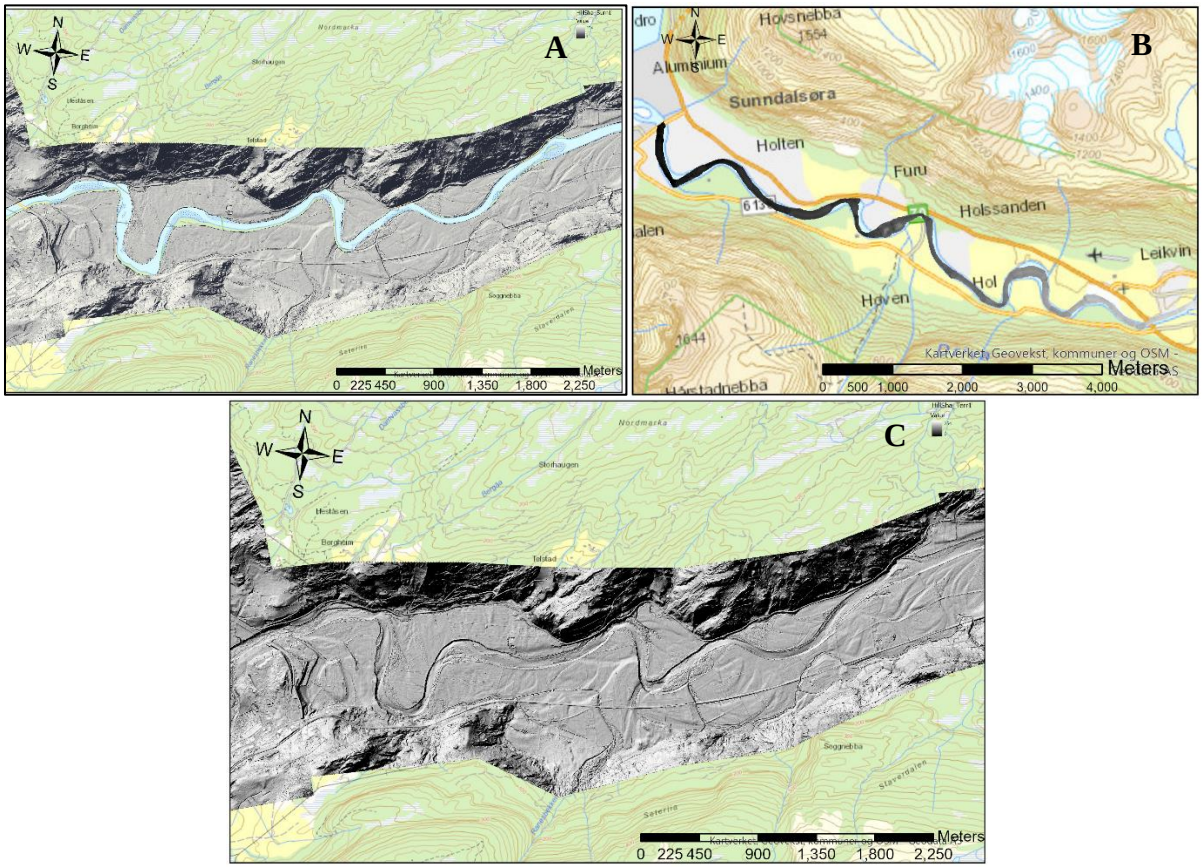


Fig 3.12 A, B, and C are the clipped Green LiDAR raster, the predicted Points raster, and the mosaiced raster

3.2.6 Artificial Neural Network Model Setup and Training Scenarios

For ANN application, API Keras code was applied for different scenarios. All the training trials were applied to the exported data from Gaula river. The data structures were Pandas dataframe and NumPy arrays for easier analysis. Jupyter Notebook platform through ANACONDA distributor was chosen for Python script writing. The three rivers later for testing were selected based on approximate sizes.

The deep learning model structure with Keras Sequential model linked with TensorFlow is created from one input dense or fully connected layer, two hidden dense layer with 164 neurons, and one output dense layer. The input and output neurons shape had a length similar to the number of points on relevant transects. For model compilation, parameters passed into the model structure are: root mean square error optimizer (rmsprop) to optimize and update the weights and learning rate to minimize the loss score through back propagation. The other parameter is the loss function selection, in this case, MSE was selected. For training arguments were passed to Keras *model.fit* which is the operator for applying the optimizer connected to the loss function to train machine learning algorithm. The data for all training trials were split

into 80% for training and 20% for validation. The shuffling argument was selected for random training.

Three different scenarios for data training have been tested to investigate the most reasonable output predicted results:

a. From Gaula 200 m transects with 400 points covering the mainstream and parts of the floodplains were considered. 100 Green LiDAR elevation points on right and left banks were fixed, forming the model input from 1200 cross-sections. The input dataframe dimension = [number of trained cross-sections ×200].
The mainstream Green LiDAR elevation points were the targeted output, represented by 200 points in the middle. The output dataframe = [number of trained cross-sections × 200]. The number of trained cross-sections was changed according to different trials.

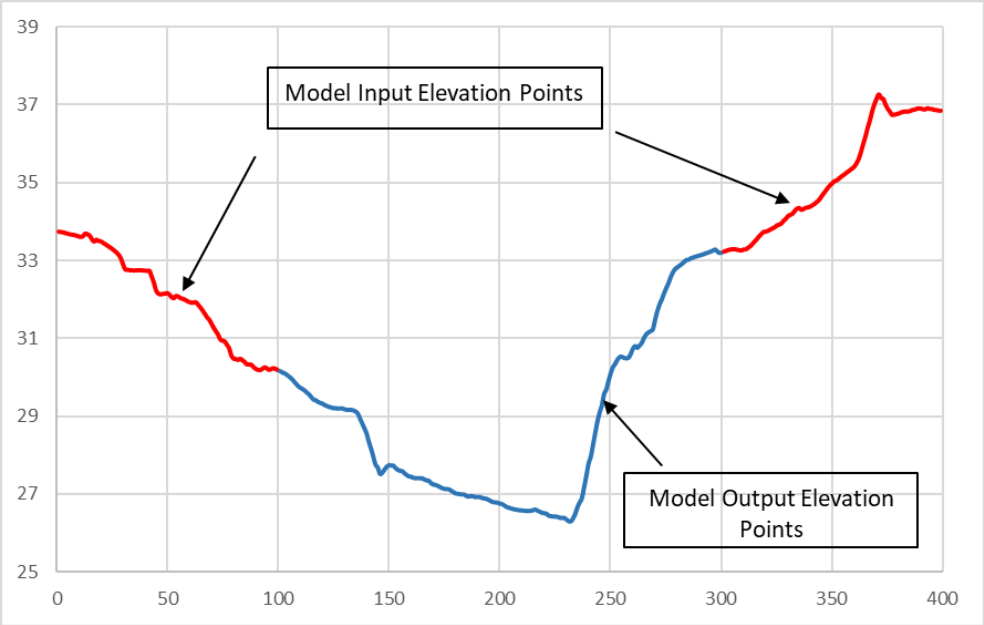


Fig 3.13 Cross Section Example of Input and Output Green LiDAR Elevation Points

b. Since the river width was varying along the stream and fixed number of points in banks and mainstream will not be representative in different parts, the second trial was trying to make the input dataframe flexible with each river cross-section width. Meanwhile, the dataframe is fully controlled by a rectangle shape. Therefore, the input and output dataframe size considered the widest cross-section along the river. To maintain the network shape for the input and the output, the empty zones in shorter cross sections filled with Zeros as Keras does not support the NaN or Na values.
c. The fourth trail was lengthening the transect for more data consideration. 400, 500, 600, and 700 m transect lengths were tested. The input shape was similar to trial c with more input points.
d. The third trial considered the full Red LiDAR elevation points as input data for the model, and the full Green LiDAR elevation points are the targeted output. Dataframe size was controlled because the transects had the same length along the whole river. The number of points varied by one point, some transects has 399 points and others has 400 points. To fit the dataframe size, one last row was added to the 399 points, the row elevation values were equal to row number 399.

3.2.7 Prediction and Testing

After the last epoch step, the model algorithm was ready for testing on different rivers. For each mentioned trial, testing was carried out on parts of Gaula where the cross-sections were not included in the deep learning training to investigate the model performance out of the training set zone. Then to generalize the algorithm application, it was also tested on the other two rivers, Driva and Surna.

3.2.8 Hydraulic Simulation

The predicted elevation points from Surna and Driva were tested in a flood model using Hydrological Engineering Center's River Analysis System HEC-RAS (version 6.0.0). The model setup was taken from a previous study for Red and Green LiDAR (Awadallah, Juárez et al. 2022). The generated raster terrain for Surna and Driva was used to run the flood simulation.

2D simulation with a 5×5 m mesh cell size was used, applying a diffusive wave equation for speed iterations and steady flow simulation. Different scenarios for flood flow hydrograph was considered for upstream boundary conditions and normal depth for downstream boundary condition considering unsteady flow simulation. A Manning's coefficient value of 0.06 was used for river banks and 0.03 for the wetted zone (Awadallah, Juárez et al. 2022). The hydraulic simulation compared the flood inundation area from the predicted bathymetry with both Green and Red LiDAR inundations using the error formula.

$$Percentage\ Error = \frac{\text{Green LiDAR inundation area} - \text{Prediction inundation area}}{\text{Green LiDAR inundation area}} \times 100\%$$

Eq 3.3

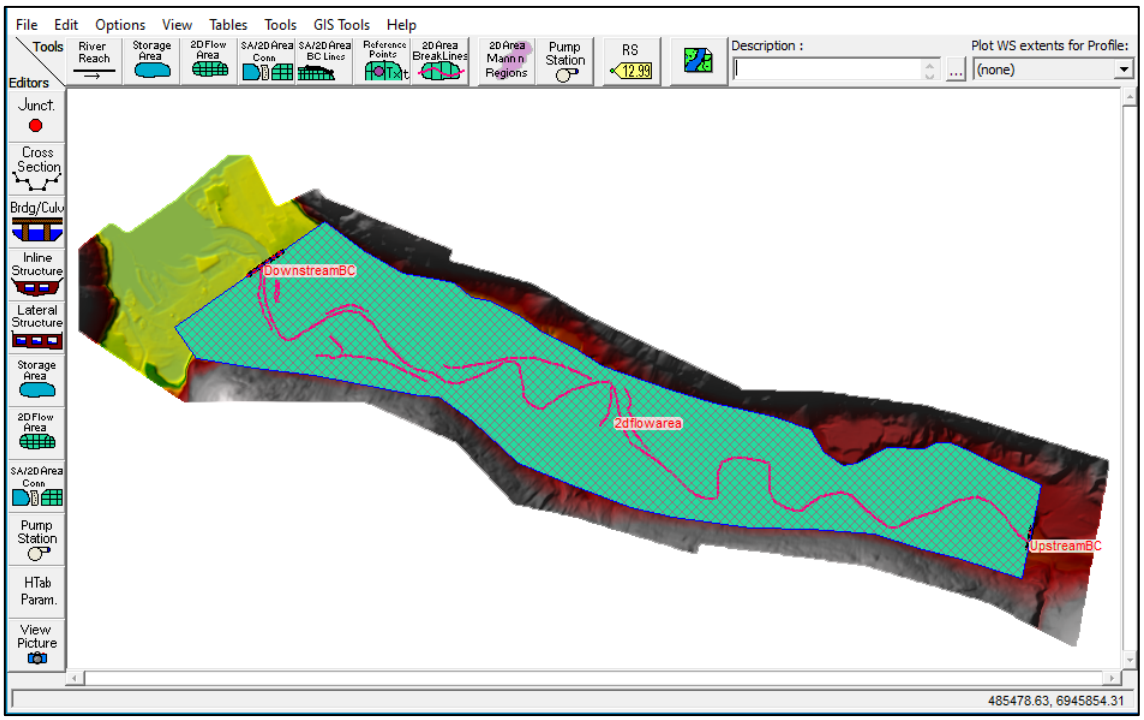


Fig 3.14 2D Flow Area and the Generated Mesh 5×5 m

Chapter 4 Results and Discussion

This section presents the results from the two machine learning methods. The first method is image inpainting, precursory test output will be illustrated and discussed. Secondly, ANN training results for trials a, b, c, and d will be shown. In addition, the flood hydraulic simulation inundation maps for mean, 10, 100, and 500 years flood discharge will be presented. More detailed graphs and maps for CNN output, ANN cross-sections training for each scenario, and Python scripts for both methods are appended to the Appendices section.

4.1 Image inpainting

The image inpainting is based on the RBG colouring index for image processing. The masked white areas were predicted with partial CNN, where the kernels filled the masked pixels depending on surrounding pixels values. Fig 4.1 shows the predicted image progress in the last two epochs. The missing pixel values were reasonably predicted compared to the targeted output image. Meanwhile, the mainstream at non-holed areas were deepened even though the pixels have existing values before the training. Higher altitudes impact is not significant, but it worth to mention the change of colours in the top left colour.

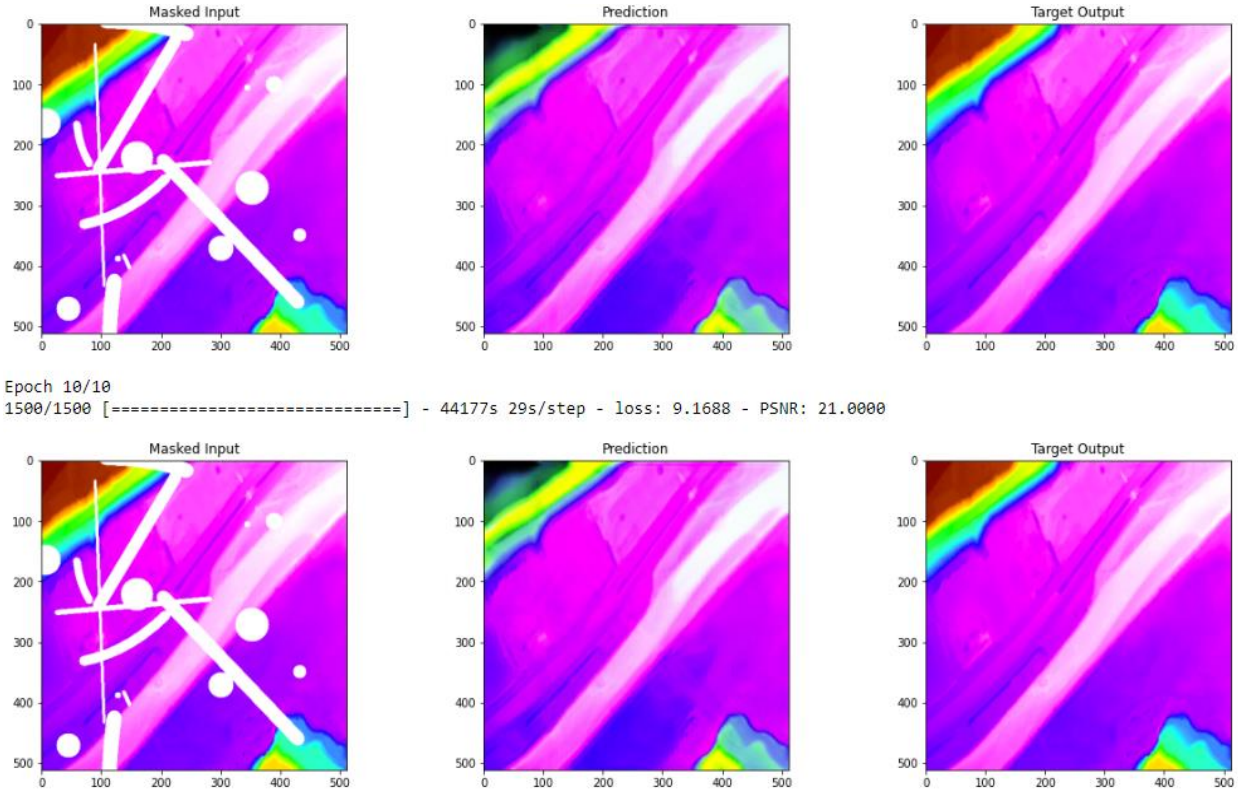


Fig 4.1 Last Two Predictions in Image Inpainting Training

Image inpainting is expected to be more successful with further testing and modifications on the targeted output and the masking type. However, training CNN on images was found to be computationally expensive and time consuming process. For instance, running the preliminary test took 8 days for completion. Moreover, data pre-processing for training was complicated and demanding task; more than 55000 masks were prepared for the training dataset, and 12000 masks for testing datasets. In return, preparing the river masks, the corresponding masked images, and the full targeted output images will be time-consuming. As the masked zones will be along the mainstream, unlike the preliminary test, more missing values exist. Thus, the algorithm requires huge amount of Green LiDAR images and masks to predict the missing data with acceptable accuracy. The method application was stopped, and we decided to move to another faster machine learning method with one-dimensional training using ANN.

4.2 1-Dimensional ANN

The training datasets were taken from Gaula in all scenarios. The remaining cross-sections, which were not included in the training, were used in testing the model performance. In addition, the models were tested in two other rivers. All the figures below illustrate results tested on sections out of the training zone in Gaula. Besides, selected results from Driva and Surna.

4.2.1 Scenario a

In this scenario, a fixed number of Green LiDAR elevation points was chosen in the banks (input data) and mainstream (targeted output) to train the model. The fixed lengths of inputs and outputs points of the cross-section was representative to some of the parts of Gaula river. The trained model in this scenario performed well for some cross-sections in which river and banks widths was similar to those used for model training. Fig 4.2 presents examples of these results which shows a slight discrepancy between the prediction and the Green LiDAR.

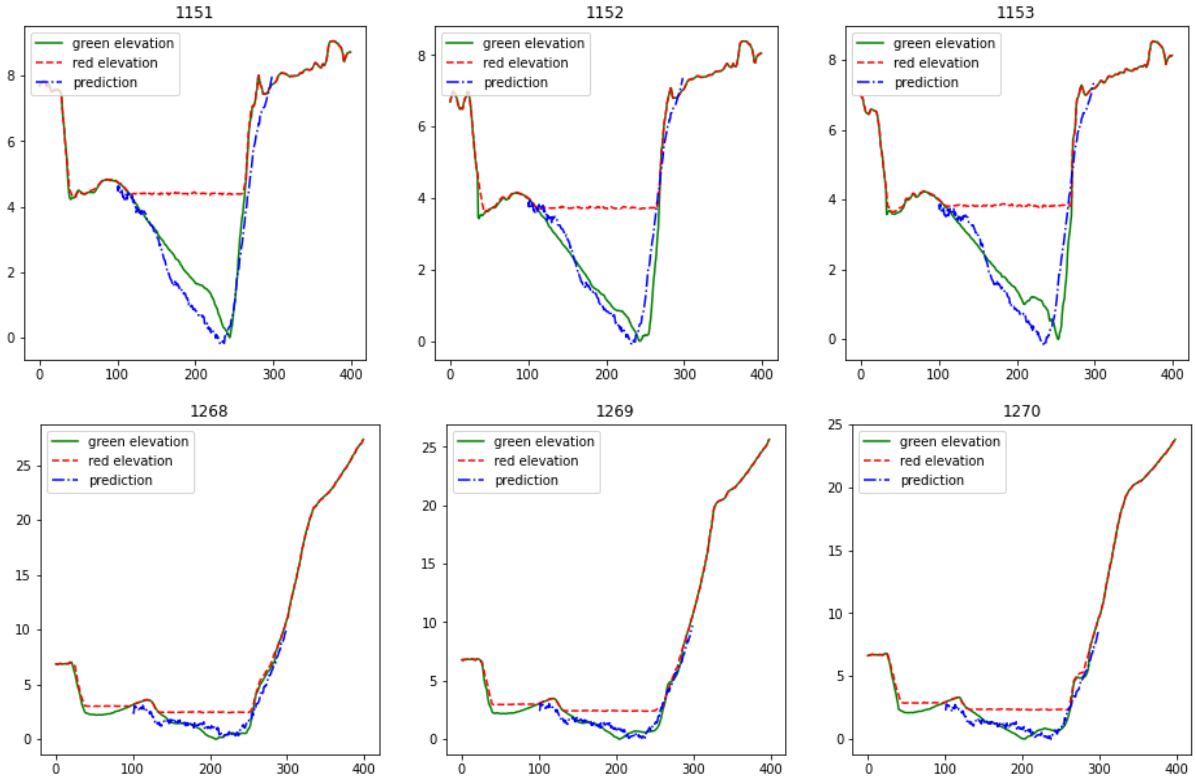


Fig 4.2 An Example of Consistency Between the Selected Number of Points and the Actual River Cross Section

In contrast, the majority of the predicted results have a large discrepancy between the prediction and the actual riverbed as illustrated in Fig 4.3. This happens for cross-sections in which the river and bank widths differ significantly from those used in model training. Therefore, selecting fixed number of points will increase the model sensitivity to the river width and banks steepness. Moreover, decreasing the model flexibility.

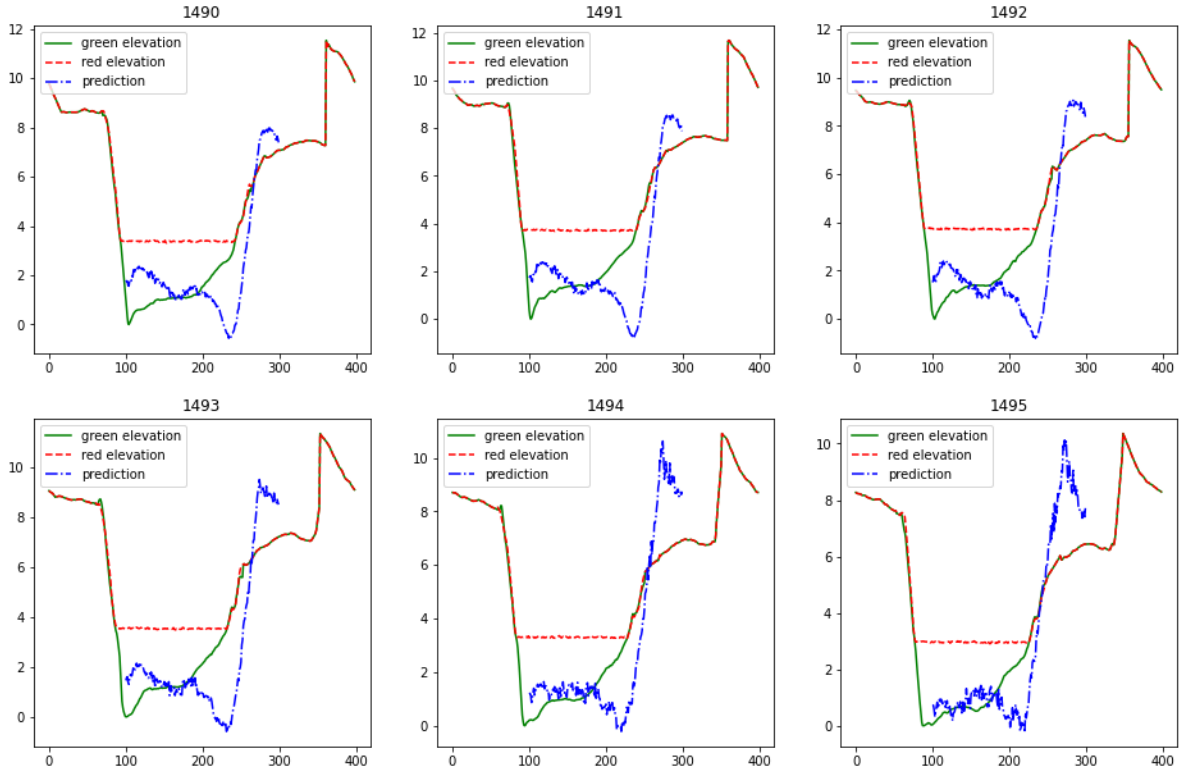


Fig 4.3 An Example of Inconsistency Between the Selected Number of Points and the Actual River Cross Section

4.2.2 Scenario b

As mentioned before inflexible model will not perform well in different river widths. Therefore, the idea of inserting changeable input and output in each river transect according to the mainstream polygon was adopted in this scenario. Where outside the polygon domain is considered input, and the points inside the polygon domain are considered as targeted output. The river width varied between 300 to 1000 points. The dataframe missing values filled with zeros to maintain the input/output structure. Keras was considering the Zeros as integer values during training, which severely affected the algorithm results as shown in Fig 4.4. The zeros were arbitrary values for maintaining the input and output dataframe shape. The method has predicted a riverbed below the Red LiDAR but with large deviation. The dataframe width is equal to the longest cross section. Hence, when predicting the narrow cross sections, the algorithm will fill the banks with zero values.

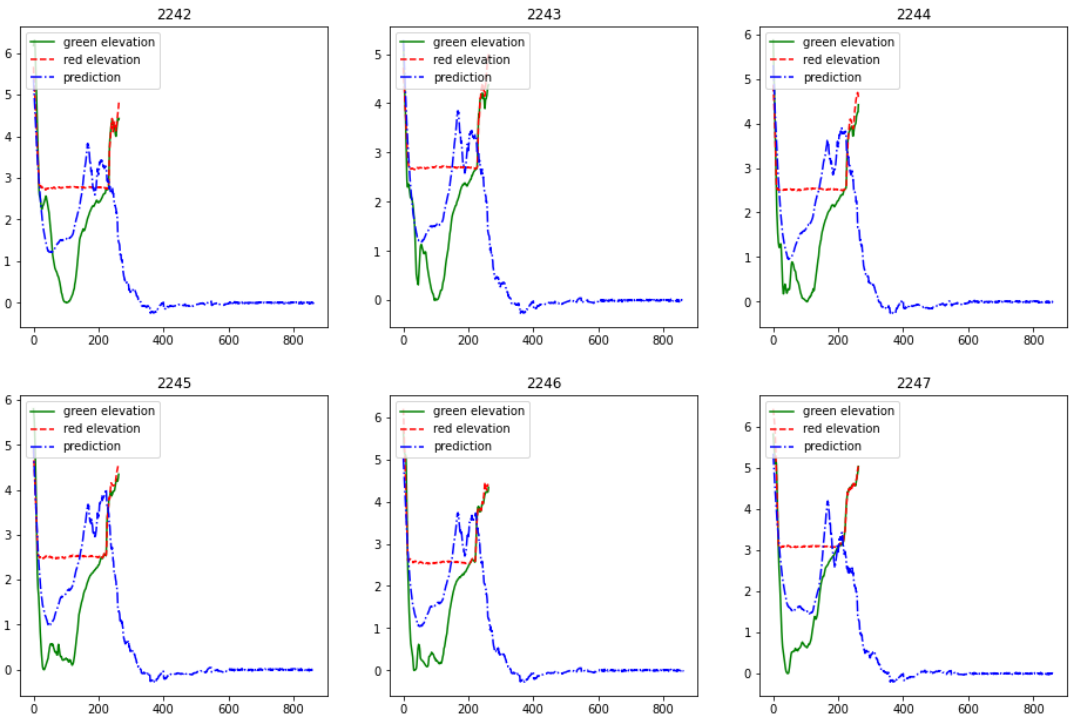


Fig 4.4 An Example of Prediction Results Using for Scenario b

4.2.3 Scenario c

Entire section training was used to overcome the gap filling issue on a dataframe. In comparison with the previous two scenarios, and complete neglection to the predicted banks, the algorithm adequately derived the predicted bathymetry on simple cross sections formations as shown in Fig 4.5 A. Nevertheless, some parts have detected deviations from the actual riverbed. On the other hand, complex formation with two or more channels or islands had a relatively large discrepancy (Fig 4.5 B). In view of the fact that the majority of training datasets was simply shaped cross sections. The model was tested in other two different rivers, Driva and Surna. The model performance was acceptable and introduced good results, despite the fact that the model was trained only in Gaula.

In this scenario, further investigation was done, Sigmoid and ReLU activation functions were tested. Two different convergence states were achieved for the same tested rivers. For example, in the Driva river (cross-section 135), the algorithm using ReLU was, to some extent, able to predict the river bathymetry on complex shape, even though the same dataset was used for the training using Sigmoid and ReLU. This highlights the importance of hyperparameter tuning for machine learning models. Hyperparameter tuning refers to the process of finding the most suitable machine learning structural parameters for the modelling task. These include activation functions, number of hidden layers, number of neurons and optimization parameters (i.e. algorithm, learning rate, momentum). Hyperparameter tuning is done by trial-and-error, or by optimization algorithms, such as genetic algorithm or Bayesian optimization. A proper hyperparameter tuning is expected to improve the accuracy of machine learning prediction. Therefore, it is recommended for future studies.

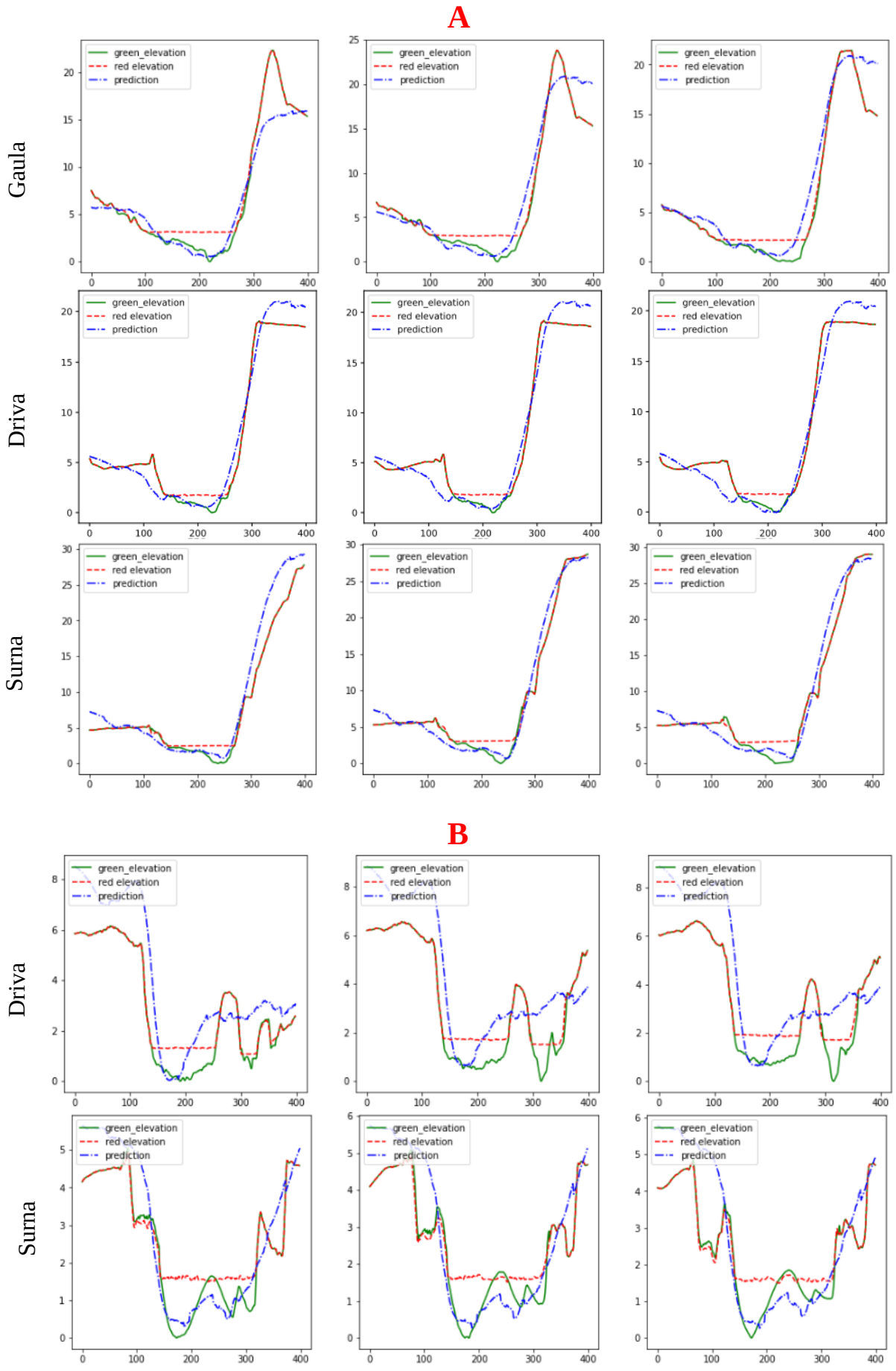


Fig 4.5 A and B Comparison Between simple(A) and Complex (B) River Channel

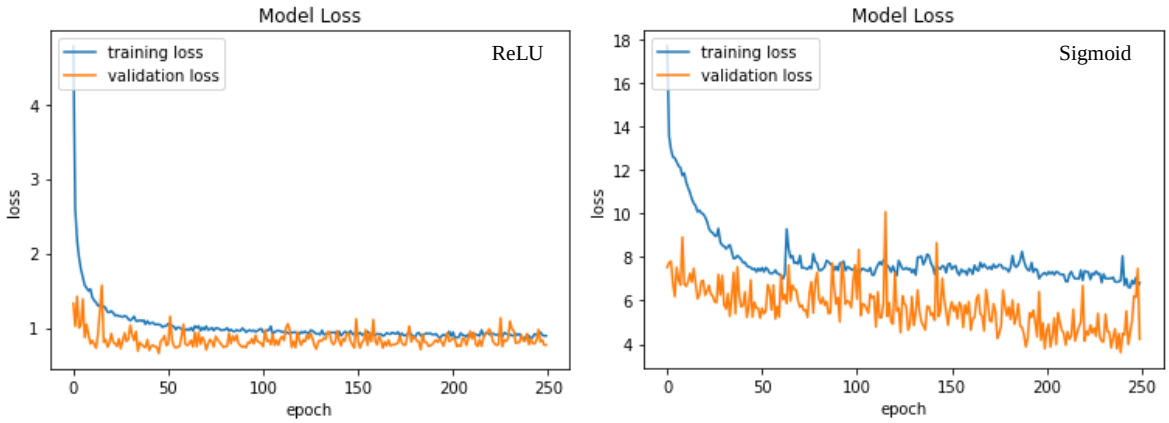


Fig 4.6 Show the MSE model loss for training and Validation

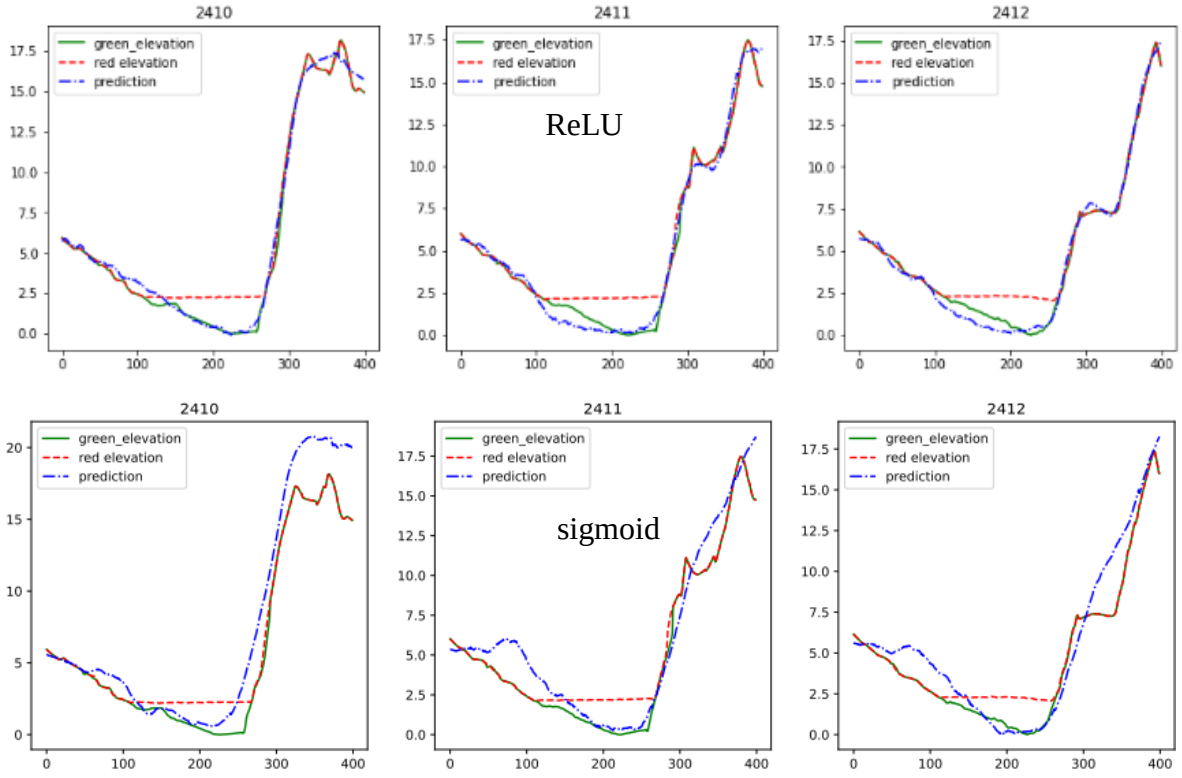


Fig 4.7 Comparison Between ReLU and Sigmoid in Cross Sections 2420,2411, and 2412, Gaula

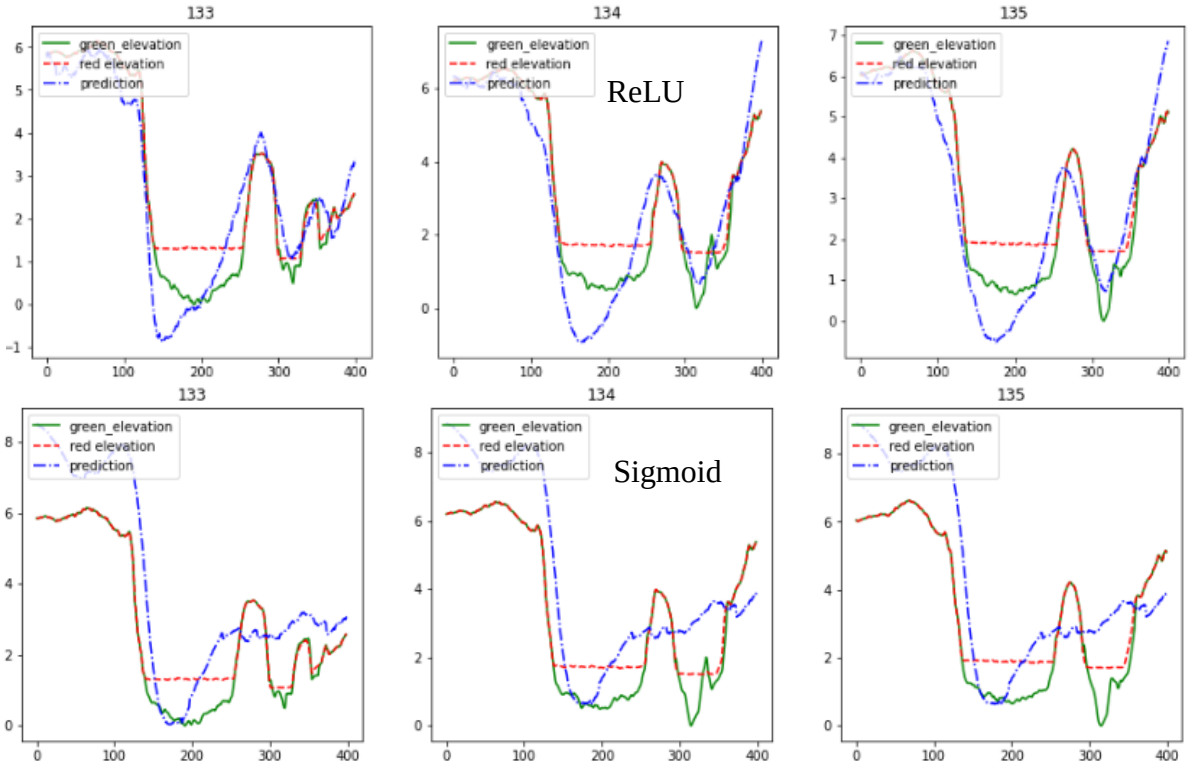


Fig 4.8 Comparison Between ReLU and Sigmoid in Cross Sections 133,134, and 135, Driva

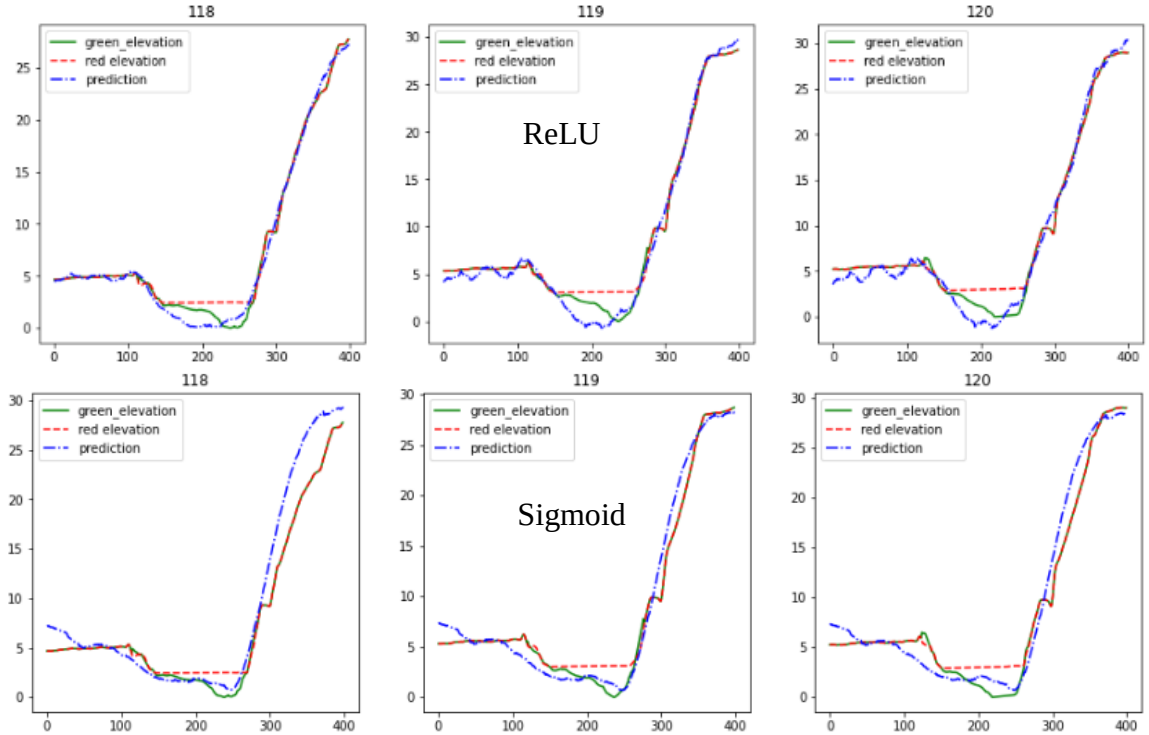


Fig 4.9 Comparison Between ReLU and Sigmoid in Cross Sections 118,119, and 120, Surna

4.2.4 Scenario d

The training output from this scenario was affected by the banks' topography. Longer transects with more elevation points were extracted to test the model demand for data. Gaula river has

quite steep banks. Consequently, the algorithm considered higher values and could not predict the river bathymetry. The banks' training was under predicting the river bottom as shown in Fig 4.10. As a result, the transect length in correspondence to the river width is a significant factor in cross-sections extraction.

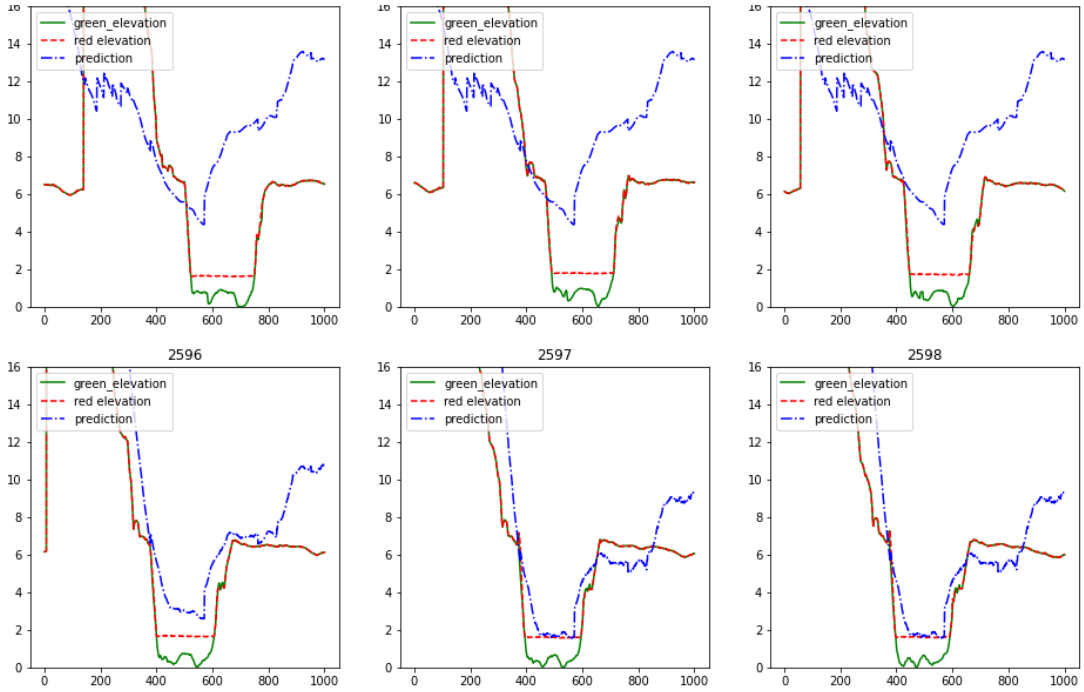


Fig 4.10 An Example of Results from Scenario d Training

4.2.5 Residual Between Green LiDAR and Predicted Bathymetry (scenario c)

The vertical difference between Green LiDAR and Prediction rasters was found using the Minus tool in ArcGis Pro. The residual magnitude was higher in positive values, which means the algorithm overpredicted the bathymetry. On the other hand, the underprediction is more distributed among the residual map, as shown in Fig 4.11. In addition, the outer river bends are underpredicted where the riverbed has sediments accumulation.Residual = Green LiDAR raster – Prediction raster

Fig 4.11 Driva Green LiDAR and Prediction Residual

4.2.6 Hydraulic Flood Simulation

Fig 4.12 illustrates the inundation areas obtained from the hydraulic flood simulation using 2D HEC-RAS in Driva. The prepared prediction DEM was used along with the Green and Red LiDAR DEM for comparison. For Driva river, flood inundation was overestimated in reaches with multiple streams (i.e. island), as shown in figure 4.11. This due to fact that ANN model failed to predict river bathymetries for section with islands, (see figure 4.5 and 4.6). On the other hand, flood inundation from the predicted terrain data matched well with those predicted by the Green lidar in reaches with single streams (i.e. without islands).

Table 4.1 compares the inundation areas generated by Green Lidar, Red lidar and the predicted terrains by the ANN. The latter yielded inundation areas with lower prediction errors than the Red LiDAR, when compared with the Green LiDAR. The error was reduced to approximately half in the mean flood scenario. While on the large flood scenarios, the improvement was 5%. In this simulation, the algorithm used to predict the geometry was built on the Sigmoid activation function. Due to time limitations, the geometry from the algorithm using ReLU was not used for flood investigation in Driva.

Table 4.2 shows the difference error in Surna. The algorithm used for the geometry prediction is based on ReLU activation function. From the presented values for all flood scenarios, the inundation area error was decreased. For example, in mean flood discharge the error decreased from 44.4% to -0.6%. However, the model overpredicted the riverbed in most cases and underestimated the flood with 8% in high flood scenarios. The model performed well in the reach where the bed formation is simple and missed the prediction in complex formations.

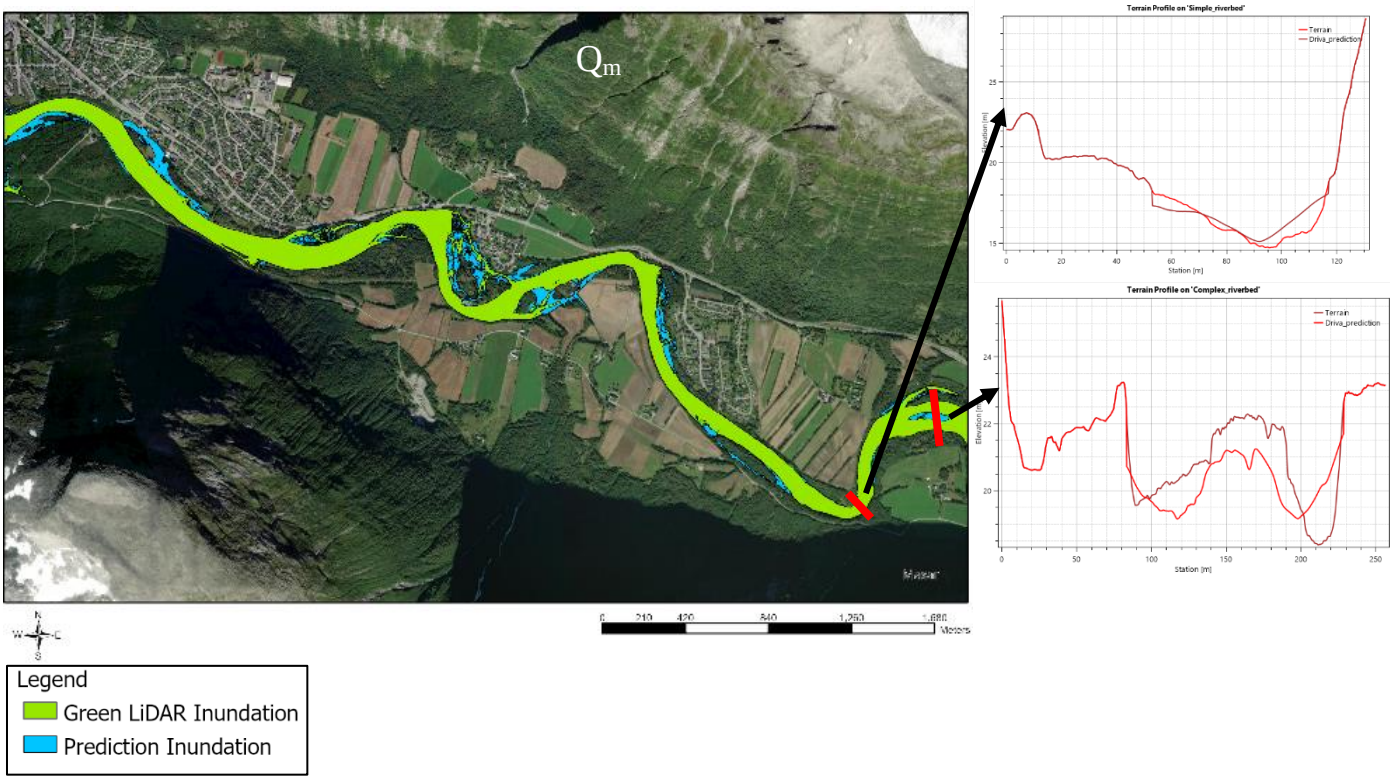


Fig 4.12 Flood Inundation Extent for Different Scenarios by HEC-RAS 6.0, Driva River

Table 4.1 Difference Error Between the Green LiDAR Vs Red LiDAR and Green LiDAR Vs Prediction, Driva/Sigmoid

Driva Flood Discharge	Green Lidar	Red Lidar	Prediction	Red Lidar Error	Prediction Error
QM	1,220,677.40	1750052.557	1,507,336.70	43.37%	23.48%
Q10	1459846.578	2059523.198	1818428.647	41.08%	24.56%
Q100	2340286.781	2909453.02	2747041.226	24.32%	17.38%
Q500	2630167.4	3190311.572	3024614.245	21.30%	15.00%

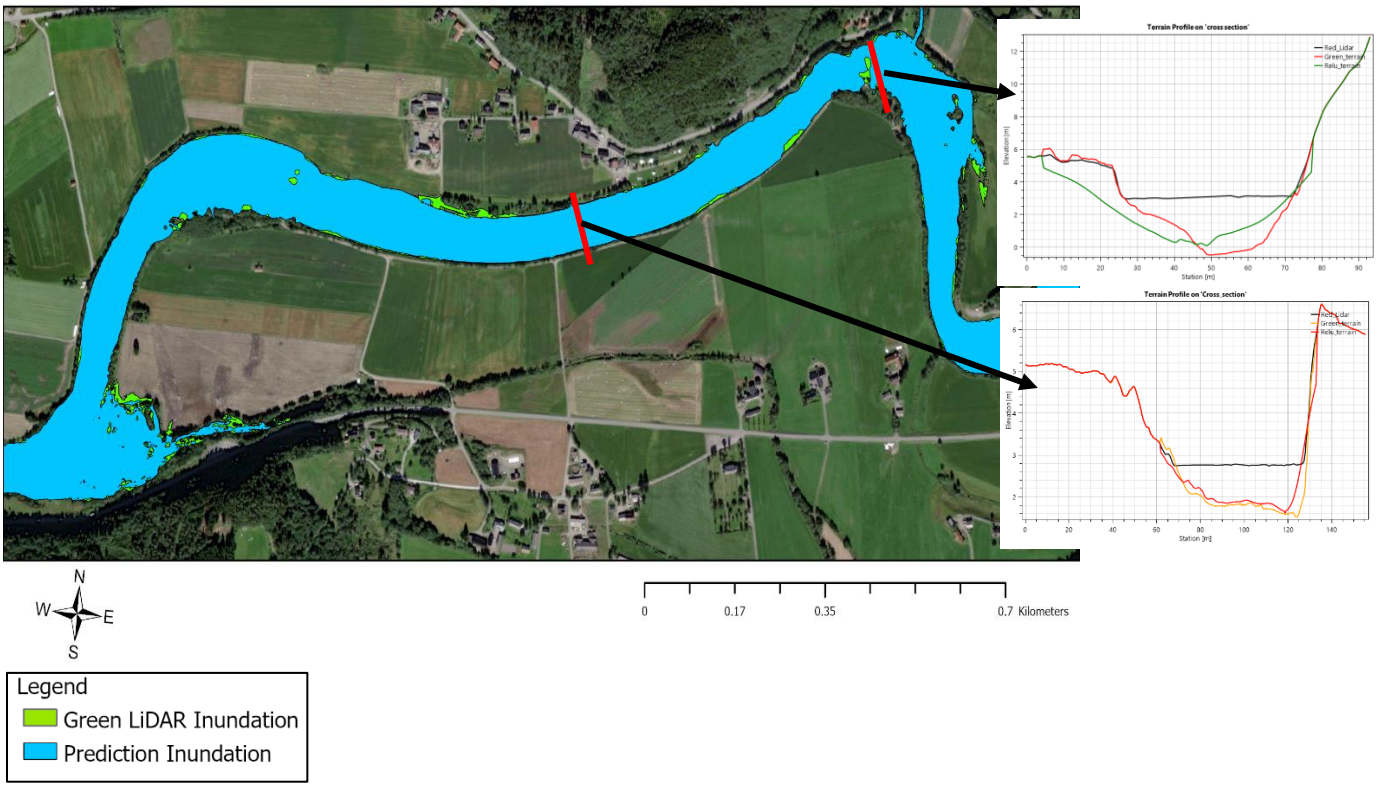


Fig 4.13 Flood Inundation Extent for Different Scenarios by HEC-RAS 6.0, Surna River

Table 4.2 Difference Error Between the Green LiDAR Vs Red LiDAR and Green LiDAR Vs Prediction, Surna/ReLU

Surna Flood Discharge	Green Lidar	Red Lidar	Prediction	Red Lidar Error	Prediction Error
QM	1,686,383.11	2434351.484	1,676,015.81	44.35%	-0.61%
Q10	2122146.106	3240414.104	2032909.973	52.70%	-4.20%
Q100	2983980.914	4452881.525	2742850.585	49.23%	-8.08%
Q500	3747158.765	5275807.478	3418006.752	40.79%	-8.78%

A study was done to obtain the river bathymetry using aerial images by (Sundt, Alfredsen et al. 2021). The study found a linear regression model using aerial images and training polygons for riverbed retrieval. The model well performed in high resolution images with and rivers with lower turbidity, and shallow river sections lower than 2 m depth.

The study limitation concluded in underestimating deeper depths in river basins, beside lower performance in low quality images and dependency on weather conditions when the images were taken and riverbanks land use status. The applied ANN machine learning method in this study is not influenced by the river depth or riverbanks vegetation and shadow. On the other hand, the regression model wasn't influenced by the existence of the island, unlike this study.

In Norway, 80% of the topography is covered by Red LiDAR survey and most of the rivers are covered, while Green LiDAR is a new and expensive technology, with limited number of covered rivers. Nevertheless, the significance of morphology and geomorphological data is increasing in river management and studying hydropeaking, sediment transport, and the ecological footprint, in addition to flood risk assessment and flood defense. This study showed promising results using ANN scenario c, reflected by the accuracy in the flood inundation area (table 4.1, 4.2). The speed of implementation and testing is fast in powerful computers such as workstation PC. The model can be transferred to other catchment areas or regions for predicting the river bathymetry. Therefore, this method is a valuable tool for river management and specifically for flood control. Another global application could be decreasing the cost by implementing the Green LiDAR survey in selected parts of the targeted river basin instead of the whole river, then using the covered areas to train a model that able to predict the remaining sections from the stream based on the Red LiDAR data.

Chapter 5 Conclusion and Recommendation

The objective of this study is to find an applicable method for obtaining river bathymetry from red lidar data. These images or DEMs are used for improving the hydraulic simulation compared to the results from Red LiDAR DEM. Two machine learning methods were applied, Image inpainting using CNN and ANN method applying Keras API code using Python. Many scenarios have been tested and evaluated for ANN. Eventually, full training for the cross-sections using the elevation points from Red LiDAR as input and Green LiDAR as targeted output from extracted points gave the best performance. The main findings from the study gave promising results on using Deep AI machine learning methods and can be summarized as follows:

- Image inpainting CNN method implied a potential in testing the method and might give high spatial resolution as it considers DEM colouring index. The need for data and the computation time was challenging with time limitations for further investigation.
- Improving the model through hyperparameter tuning, more factors consideration, and training datasets would increase the algorithm prediction accuracy with specified uncertainties. Therefore, reasonable predictions in complex cross-sections and different river sizes.
- Two activation functions were tested on the model training. The model architecture is input layer, 2 hidden layers and output layer. ReLU and Sigmoid were applied in the two hidden layers, the output prediction from ReLU activation function model was more accurate than Sigmoid.
- The resulted algorithm from the ReLU model was tested in Surna geometry. Hydraulic simulation was done to find the inundation areas with Green LiDAR, it varied between 0.6 % to 9% less inundation area (overpredicted riverbed).

To conclude, machine learning (ML) is a very robust and flexible tool for data-driven models. It can be modified and investigated by adjusting the datasets. The drawback is that ML is a black box model, which will need trial and error to achieve the best performance. To parametrize and weight the input factors significance, it will require time-intense mathematical computations

Future recommendations for the two machine learning methods:

- Firstly, image inpainting method: the colouring scheme for the mainstream in the targeted output has a light colour gradient, this might disturb the kernel between masked and unmasked zones, as the masked zones have a white colour index. Therefore, it affects the partial CNN functionality. Changing the colouring scheme to avoid the kernel confusion might decrease the effect of filters targeting the non-holed area, and maintaining the original mainstream pixel values.
- Transforming the mask from randomly generated mask over the entire image, to masks covering the mainstream to train the model. The targeted output should be enough number of RBG images along the trained river, toward an algorithm that is able to predict the missing values on other testing streams.

- Secondly, the ANN model: hyperparameter tuning and model selection to avoid overfitting while testing on other rivers by examining different hyperparameters settings such as; the number of epochs, batch size, optimizer, activation function, etc. it is recommended to apply the hyperparameter tuning in scenario c.
- A new scenario could be, training the model based on the wetted zone only excluding the banks and fully depend on the bathymetry. Considering the Red LiDAR elevation points on the wetted zone as input, beside the Green LiDAR elevation points as targeted output. This will help to avoid the model confusion by diverse banks steepness. Again, maintain the dataframe shape will be an issue; this could be solved by masking the inserted values to keep the dataframe structure.
- In this study only Gaula was considered for the model training. Other different rivers in Norway have high-quality bathymetry information could be added to the trained model. This will include varied cross sections formations and sizes, to investigate the model capability of predicting different sizes. For example, if the model is able to predict islands existence, this will significantly improve the flood simulation
- Another different factors impact can be studied, such as including the river discharge, banks slope in case of full sections training, and do some uncertainty analysis.
- Although the flood inundation areas are more reasonable in comparison to Red LiDAR, but the water velocity was not investigated in this study.

References

Abdalla, E. M. H., et al. (2021). "Evaluating different machine learning methods to simulate runoff from extensive green roofs." Hydrol. Earth Syst. Sci. **25**(11): 5917-5935.

Abdi, G., et al. (2018). "Deep learning decision fusion for the classification of urban remote sensing data." Journal of Applied Remote Sensing **12**(1): 016038.

Alaghmand, S., et al. (2012). "Comparison between capabilities of HEC-RAS and MIKE11 hydraulic models in river flood risk modeling (a case study of Sungai Kayu Ara River basin, Malaysia)." International Journal of Hydrology Science and Technology **2**: 270-291.

Albawi, S., et al. (2017). Understanding of a convolutional neural network. 2017 international conference on engineering and technology (ICET), Ieee.

Algretawee, H. and G. Alshama (2021). "Modeling of Evapotranspiration (ETo) in a Medium Urban Park within a Megacity by Using Artificial Neural Network (ANN) Model." Periodica Polytechnica Civil Engineering **65**(4): 1260-1268.

Allouis, T., et al. (2010). "Comparison of LiDAR waveform processing methods for very shallow water bathymetry using Raman, near-infrared and green signals." Earth Surface Processes and Landforms: The Journal of the British Geomorphological Research Group **35**(6): 640-650.

Awadallah, M. O. M., et al. (2022). "Comparison between Topographic and Bathymetric LiDAR Terrain Models in Flood Inundation Estimations." Remote Sensing **14**(1): 227.

Bales, J. D., et al. (2007). LiDAR-derived flood-inundation maps for real-time flood-mapping applications, Tar River Basin, North Carolina, Geological Survey (US).

Bertalmio, M., et al. (2000). Image inpainting. Proceedings of the 27th annual conference on Computer graphics and interactive techniques.

Bird, S., et al. (2010). "Photogrammetric monitoring of small streams under a riparian forest canopy." Earth surface processes and landforms **35**(8): 952-970.

Bloschl, G., et al. (2017). "Changing climate shifts timing of European floods." Science **357**(6351): 588-590.

Brasington, J., et al. (2000). "Monitoring and modelling morphological change in a braided gravel-bed river using high resolution GPS-based survey." Earth Surface Processes and Landforms: The Journal of the British Geomorphological Research Group **25**(9): 973-990.

Casas, A., et al. (2006). "The topographic data source of digital terrain models as a key element in the accuracy of hydraulic flood modelling." Earth Surface Processes and Landforms: The Journal of the British Geomorphological Research Group **31**(4): 444-456.

Chandler, J. (1999). "Effective application of automated digital photogrammetry for geomorphological research." Earth surface processes and landforms **24**(1): 51-63.

Chen, B., et al. (2017). "Using LiDAR surveys to document floods: A case study of the 2008 Iowa flood." Journal of Hydrology **553**: 338-349.

Cheng, S., et al. (2021). "Machine learning for predicting discharge fluctuation of a karst spring in North China." Acta Geophysica **69**(1): 257-270.

Das, K. and R. N. Behera (2017). "A survey on machine learning: concept, algorithms and applications." International Journal of Innovative Research in Computer and Communication Engineering **5**(2): 1301-1309.

Deparday, V., et al. (2019). "Machine learning for disaster risk management."

Dowman, I. (2004). "Integration of LIDAR and IFSAR for mapping." International Archives of Photogrammetry and Remote Sensing **35**(B2): 90-100.

Feng, S., et al. (2018). "Using deep neural network with small dataset to predict material defects." Materials & Design **162**.

Fonstad, M. A. and W. A. Marcus (2010). "High resolution, basin extent observations and implications for understanding river form and process." Earth Surface Processes and Landforms: The Journal of the British Geomorphological Research Group **35**(6): 680-698.

Gao, J. (2009). "Bathymetric mapping by means of remote sensing: methods, accuracy and limitations." Progress in Physical Geography **33**(1): 103-116.

Hodgson, M. E., et al. (2003). "An evaluation of LIDAR-and IFSAR-derived digital elevation models in leaf-on conditions with USGS Level 1 and Level 2 DEMs." Remote sensing of environment **84**(2): 295-308.

Jinlong, Z. and Y. JinTao (2009). Application of neural network in groundwater denitrification process. 2009 Asia-Pacific Conference on Information Processing, IEEE.

Khaniani, A. S., et al. (2021). "Rainfall forecast based on GPS PWV together with meteorological parameters using neural network models." Journal of Atmospheric and Solar-Terrestrial Physics **214**: 105533.

Kinzel, P. J., et al. (2013). "Mapping river bathymetry with a small footprint green LiDAR: applications and challenges 1." JAWRA Journal of the American Water Resources Association **49**(1): 183-204.

Kraus, K. and N. Pfeifer (1998). "Determination of terrain models in wooded areas with airborne laser scanner data." ISPRS Journal of Photogrammetry and Remote Sensing **53**(4): 193-203.

Kron, W., et al. (2019). "Reduction of flood risk in Europe - Reflections from a reinsurance perspective." Journal of Hydrology **576**: 197-209.

Kulisz, M., et al. (2021). "Forecasting water quality index in groundwater using artificial neural network." Energies **14**(18): 5875.

Lane, S., et al. (2000). "Application of digital photogrammetry to complex topography for geomorphological research." The Photogrammetric Record **16**(95): 793-821.

Lane, S., et al. (1993). "Developments in photogrammetry; the geomorphological potential." Progress in Physical Geography **17**(3): 306-328.

Liu, G., et al. (2018). Image inpainting for irregular holes using partial convolutions. Proceedings of the European conference on computer vision (ECCV).

Machado, F., et al. (2011). "Monthly rainfall–runoff modelling using artificial neural networks." Hydrological Sciences Journal–Journal des Sciences Hydrologiques **56**(3): 349-361.

MathiasGruber (2019). "Image Inpainting Code." from https://github.com/MathiasGruber/PConv-Keras.git.

Meng, X., et al. (2010). "Ground filtering algorithms for airborne LiDAR data: A review of critical issues." Remote Sensing **2**(3): 833-860.

Meng, X., et al. (2009). "A multi-directional ground filtering algorithm for airborne LIDAR." ISPRS Journal of Photogrammetry and Remote Sensing **64**(1): 117-124.

Mercer, B. (2004). DEMs created from airborne IFSAR–an update. Proceedings of the ISPRS XXth Congress.

Montesinos-López, O., et al. (2022). Fundamentals of Artificial Neural Networks and Deep Learning**:** 379-425.

Muhadi, N. A., et al. (2020). "The use of LiDAR-derived DEM in flood applications: A review." Remote Sensing **12**(14): 2308.

O'Shea, K. and R. Nash (2015). "An introduction to convolutional neural networks." arXiv preprint arXiv:1511.08458.

Qi, Y., et al. (2019). "A decomposition-ensemble learning model based on LSTM neural network for daily reservoir inflow forecasting." Water Resources Management **33**(12): 4123-4139.

Roshni, T., et al. (2020). "Neural network modeling for groundwater-level forecasting in coastal aquifers." Neural Computing and Applications **32**(16): 12737-12754.

Saha, S. (2018). "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way." from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

Sampson, C. C., et al. (2012). "Use of terrestrial laser scanning data to drive decimetric resolution urban inundation models." Advances in water resources **41**: 1-17.

Santillan, J. R., et al. (2016). Integrating LiDAR and flood simulation models in determining exposure and vulnerability of buildings to extreme rainfall-induced flood hazards. 2016 IEEE international geoscience and remote sensing symposium (IGARSS), IEEE.

Semiromi, M. T., et al. (2018). "Reducing computational costs of automatic calibration of rainfall-runoff models: meta-models or high-performance computers?" Water **10**(10): 1440.

Sharma, S., et al. (2017). "Activation functions in neural networks." towards data science **6**(12): 310-316.

Shigidi, A. and L. A. Garcia (2003). "Parameter estimation in groundwater hydrology using artificial neural networks." Journal of computing in civil engineering **17**(4): 281-289.

Shrestha, R. and F. Nestmann (2009). "Physically Based and Data-Driven Models and Propagation of Input Uncertainties in River Flood Prediction." Journal of Hydrologic Engineering **14**: 1309-1319.

Sivastava, A. K., et al. (2022). "Neural Network based Predictors for Evaporation Estimation at Jabalpur in Central India." Journal of Scientific and Industrial Research (JSIR) **81**(03): 319-328.

Sreekanth, P., et al. (2009). "Forecasting groundwater level using artificial neural networks." Current science: 933-939.

Stevenson, J. A., et al. (2010). "Despeckling SRTM and other topographic data with a denoising algorithm." Geomorphology **114**(3): 238-252.

Stylianoudaki, C., et al. (2022). "Modeling groundwater nitrate contamination using artificial neural networks." Water **14**(7): 1173.

Sundt, H., et al. (2021). "Regionalized Linear Models for River Depth Retrieval Using 3-Band Multispectral Imagery and Green LIDAR Data." Remote Sensing **13**(19): 3897.

TensorFlow (2022). "Keras Sequential ". from https://www.tensorflow.org/guide/keras/sequential_model.

Wang, Q., et al. (2020). "Sequence-based statistical downscaling and its application to hydrologic simulations based on machine learning and big data." Journal of Hydrology **586**: 124875.

Weber, T., et al. (2020). "Deep learning for creating surrogate models of precipitation in Earth system models." Atmospheric Chemistry and Physics **20**(4): 2303-2317.

Webster, T. L., et al. (2004). "Using topographic lidar to map flood risk from storm-surge events for Charlottetown, Prince Edward Island, Canada." Canadian Journal of Remote Sensing **30**(1): 64-76.

Wendi, D., et al. (2016). "An innovative approach to improve SRTM DEM using multispectral imagery and artificial neural network." Journal of Advances in Modeling Earth Systems **8**(2): 691-702.

Yegnanarayana, B. (2009). Artificial neural networks, PHI Learning Pvt. Ltd.

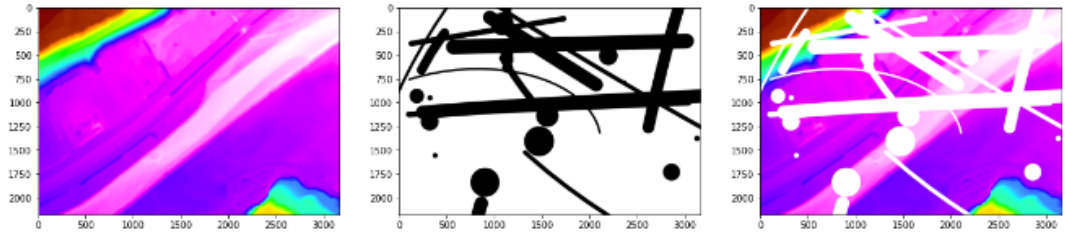
Zhang, L., et al. (2016). "Deep learning for remote sensing data: A technical tutorial on the state of the art." IEEE Geoscience and remote sensing magazine **4**(2): 22-40.

Appendices

This code for image inpainting is developed by (MathiasGruber 2019)

In [45]:
```
# Load image
img = cv2.imread('C:/Users/raffaa/Desktop/River_mask/Sokna.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
shape = img.shape
print(f"Shape of image is: {shape}")

# Instantiate mask generator
mask_generator = MaskGenerator(shape[0], shape[1], 3, rand_seed=42)

# Load mask
mask = mask_generator.sample()

# Image + mask
masked_img = deepcopy(img)
masked_img[mask==0] = 255

# Show side by side
_, axes = plt.subplots(1, 3, figsize=(20, 5))
axes[0].imshow(img)
axes[1].imshow(mask*255)
axes[2].imshow(masked_img)
plt.show()
```

```
Shape of image is: (2180, 3164, 3)
```



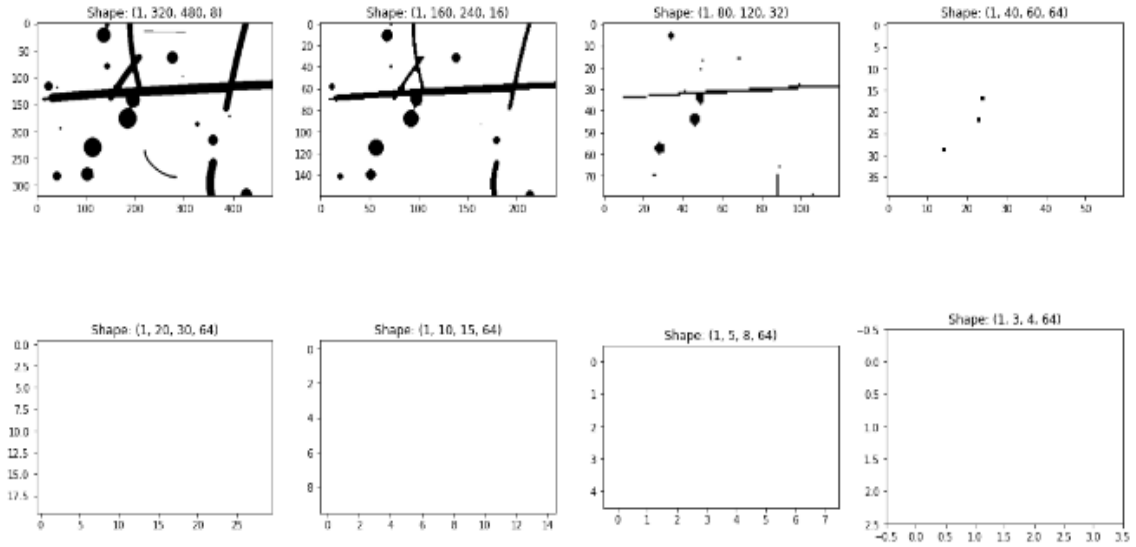
In [46]:
```
from Lidpack.pconv_layer import PConv2D

# Input images and masks
input_img = Input(shape=(shape[0], shape[1], shape[2],))
input_mask = Input(shape=(shape[0], shape[1], shape[2],))
output_img, output_mask1 = PConv2D(8, kernel_size=(7,7), strides=(2,2))([input_img, input_mask])
output_img, output_mask2 = PConv2D(16, kernel_size=(5,5), strides=(2,2))([output_img, output_mask1])
output_img, output_mask3 = PConv2D(32, kernel_size=(5,5), strides=(2,2))([output_img, output_mask2])
output_img, output_mask4 = PConv2D(64, kernel_size=(3,3), strides=(2,2))([output_img, output_mask3])
output_img, output_mask5 = PConv2D(64, kernel_size=(3,3), strides=(2,2))([output_img, output_mask4])
output_img, output_mask6 = PConv2D(64, kernel_size=(3,3), strides=(2,2))([output_img, output_mask5])
output_img, output_mask7 = PConv2D(64, kernel_size=(3,3), strides=(2,2))([output_img, output_mask6])
output_img, output_mask8 = PConv2D(64, kernel_size=(3,3), strides=(2,2))([output_img, output_mask7])

# Create model
model = Model(
    inputs=[input_img, input_mask],
    outputs=[
        output_img, output_mask1, output_mask2,
        output_mask3, output_mask4, output_mask5,
        output_mask6, output_mask7, output_mask8
    ])
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])

# Show summary of the model
model.summary()
```

```
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
input_17 (InputLayer)        (None, 2180, 3164, 3 0
_________________________________________________________________________________________________
input_18 (InputLayer)        (None, 2180, 3164, 3 0
_________________________________________________________________________________________________
p_conv2d_97 (PConv2D)        [(None, 1090, 1582,  1184        input_17[0][0]
                                                              input_18[0][0]
_________________________________________________________________________________________________
p_conv2d_98 (PConv2D)        [(None, 545, 791, 16 3216        p_conv2d_97[0][0]
                                                              p_conv2d_97[0][1]
_________________________________________________________________________________________________
p_conv2d_99 (PConv2D)        [(None, 273, 396, 32 12832       p_conv2d_98[0][0]
                                                              p_conv2d_98[0][1]
_________________________________________________________________________________________________
```

In [47]:
```
formatted_img = np.expand_dims(masked_img, 0) / 255
formatted_mask = np.expand_dims(mask, 0)
print(f"Original Mask Shape: {formatted_mask.shape} - Max value in mask: {np.max(formatted_mask)}")

output_img, o1, o2, o3, o4, o5, o6, o7, o8 = model.predict([formatted_img, formatted_mask])
```

```
Original Mask Shape: (1, 2180, 3164, 3) - Max value in mask: 1
```

In [37]:
```
_, axes = plt.subplots(2, 4, figsize=(20, 10))
axes[0][0].imshow(o1[0, :,:, 0], cmap = 'gray', vmin=0, vmax=1)
axes[0][1].imshow(o2[0, :,:, 0], cmap = 'gray', vmin=0, vmax=1)
axes[0][2].imshow(o3[0, :,:, 0], cmap = 'gray', vmin=0, vmax=1)
axes[0][3].imshow(o4[0, :,:, 0], cmap = 'gray', vmin=0, vmax=1)
axes[1][0].imshow(o5[0, :,:, 0], cmap = 'gray', vmin=0, vmax=1)
axes[1][1].imshow(o6[0, :,:, 0], cmap = 'gray', vmin=0, vmax=1)
axes[1][2].imshow(o7[0, :,:, 0], cmap = 'gray', vmin=0, vmax=1)
axes[1][3].imshow(o8[0, :,:, 0], cmap = 'gray', vmin=0, vmax=1)
axes[0][0].set_title(f"Shape: {o1.shape}")
axes[0][1].set_title(f"Shape: {o2.shape}")
axes[0][2].set_title(f"Shape: {o3.shape}")
axes[0][3].set_title(f"Shape: {o4.shape}")
axes[1][0].set_title(f"Shape: {o5.shape}")
axes[1][1].set_title(f"Shape: {o6.shape}")
axes[1][2].set_title(f"Shape: {o7.shape}")
axes[1][3].set_title(f"Shape: {o8.shape}")
plt.show()
```



In [48]:
```
import os
import gc
import copy

import numpy as np
from PIL import Image

from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import TensorBoard, ModelCheckpoint, LambdaCallback

import matplotlib
import matplotlib.pyplot as plt

# Change to root path
if os.path.basename(os.getcwd()) != 'PConv-Keras':
    os.chdir('..')

from libpack.util import MaskGenerator
from libpack.pconv_model import PConvUnet

# Settings
BATCH_SIZE = 4

# Imagenet Rescaling
MEAN = np.array([0.485, 0.456, 0.406])
STD = np.array([0.229, 0.224, 0.225])

%matplotlib inline
%load_ext autoreload
%autoreload 2
```

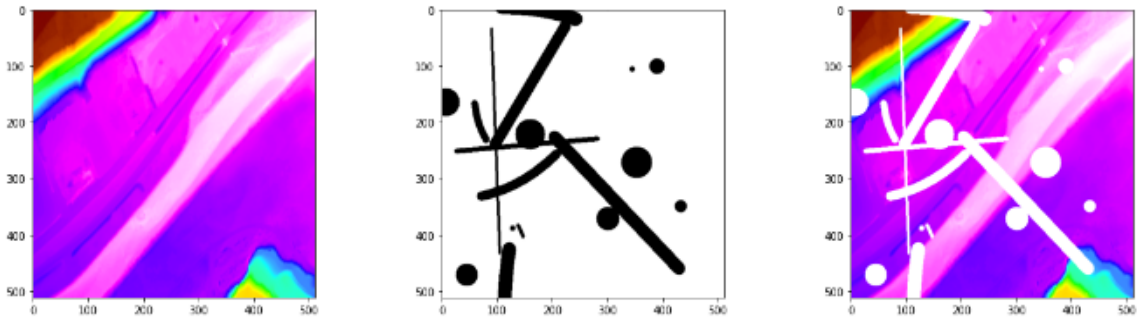
```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

In [49]:
```
PConvUnet().summary()
```

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
inputs_img (InputLayer)         (None, 512, 512, 3)  0
__________________________________________________________________________________________________
inputs_mask (InputLayer)        (None, 512, 512, 3)  0
__________________________________________________________________________________________________
p_conv2d_105 (PConv2D)          [(None, 256, 256, 64 9472        inputs_img[0][0]
                                                                 inputs_mask[0][0]
__________________________________________________________________________________________________
activation_33 (Activation)      (None, 256, 256, 64) 0           p_conv2d_105[0][0]
__________________________________________________________________________________________________
p_conv2d_106 (PConv2D)          [(None, 128, 128, 12 204928      activation_33[0][0]
                                                                 p_conv2d_105[0][1]
__________________________________________________________________________________________________
EncBN1 (BatchNormalization)     (None, 128, 128, 128 512         p_conv2d_106[0][0]
__________________________________________________________________________________________________
activation_34 (Activation)      (None, 128, 128, 128 0           EncBN1[0][0]
__________________________________________________________________________________________________
```

Testing out on single image

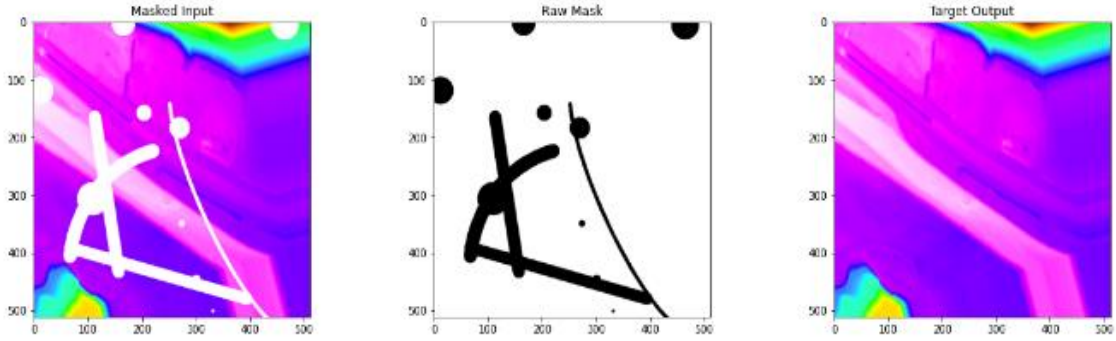
In [50]:
```
# Instantiate mask generator
mask_generator = MaskGenerator(512, 512, 3, rand_seed=42)

# Load image
img = np.array(Image.open('C:/Users/raffaa/Desktop/River_mask/Sokna.jpg').resize((512, 512))) / 255

# Load mask
mask = mask_generator.sample()

# Image + mask
masked_img = copy.deepcopy(img)
masked_img[mask==0] = 1

# Show side by side
_, axes = plt.subplots(1, 3, figsize=(20, 5))
axes[0].imshow(img)
axes[1].imshow(mask*255)
axes[2].imshow(masked_img)
plt.show()
```



Creating data generator

In [51]:
```
def plot_sample_data(masked, mask, ori, middle_title='Raw Mask'):
    _, axes = plt.subplots(1, 3, figsize=(20, 5))
    axes[0].imshow(masked[:,:,:])
    axes[0].set_title('Masked Input')
    axes[1].imshow(mask[:,:,:])
    axes[1].set_title(middle_title)
    axes[2].imshow(ori[:,:,:])
    axes[2].set_title('Target Output')
    plt.show()

class DataGenerator(ImageDataGenerator):
    def flow(self, x, *args, **kwargs):
        while True:

            # Get augmentend image samples
            ori = next(super().flow(x, *args, **kwargs))

            # Get masks for each image sample
            mask = np.stack([mask_generator.sample() for _ in range(ori.shape[0])], axis=0)

            # Apply masks to all image sample
            masked = copy.deepcopy(ori)
            masked[mask==0] = 1

            # Yield ([ori, mask],  ori) training batches
            # print(masked.shape, ori.shape)
            gc.collect()
            yield [masked, mask], ori

# Create datagen
datagen = DataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True
)

# Create generator from numpy array
batch = np.stack([img for _ in range(BATCH_SIZE)], axis=0)
generator = datagen.flow(x=batch, batch_size=BATCH_SIZE)
```

In [52]:
```
[m1, m2], o1 = next(generator)
plot_sample_data(m1[0], m2[0]*255, o1[0])
```

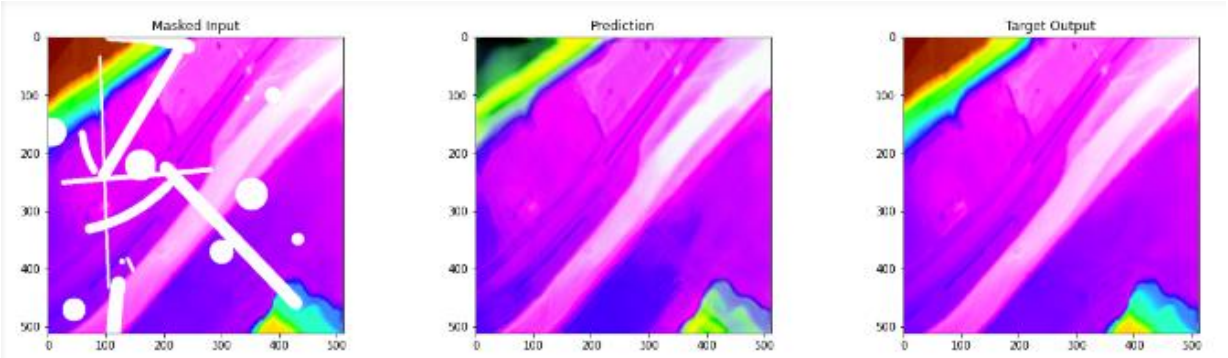


Training the inpainting UNet on single image

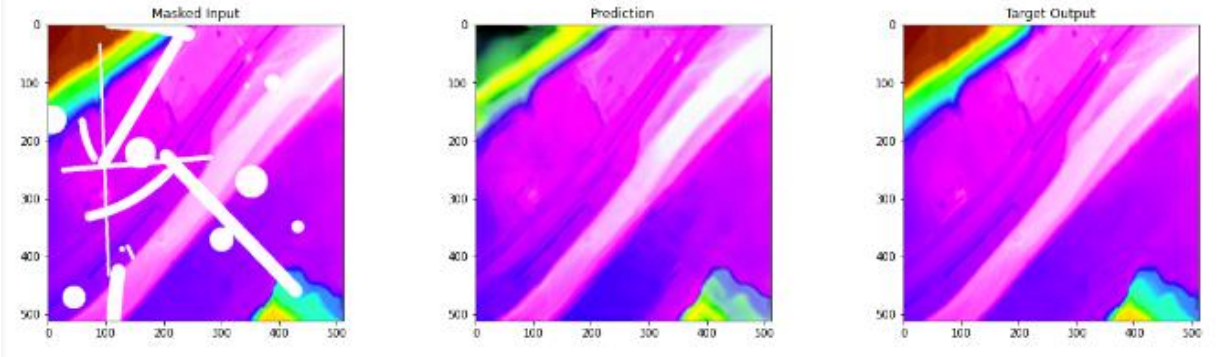
In [53]:
```
model = PConvUnet(vgg_weights="imagenet")
```

In [54]:
```
model.fit_generator(
    generator,
    steps_per_epoch=1500,
    epochs=10,
    callbacks=[
        TensorBoard(
            log_dir='C:/Users/raffaa/Desktop/River_masks/River_train',
            write_graph=False
        ),
        ModelCheckpoint(
            'C:/Users/raffaa/Desktop/River_masks/River_train/weights.{epoch:02d}-{loss:.2f}.h5',
            monitor='loss',
            save_best_only=True,
            save_weights_only=True
        ),
        LambdaCallback(
            on_epoch_end=lambda epoch, logs: plot_sample_data(
                masked_img,
                model.predict(
                    [
                        np.expand_dims(masked_img,0),
                        np.expand_dims(mask,0)
                    ]
                )[0]
                ,
                img,
                middle_title='Prediction'
            )
        )
    ],
)
```

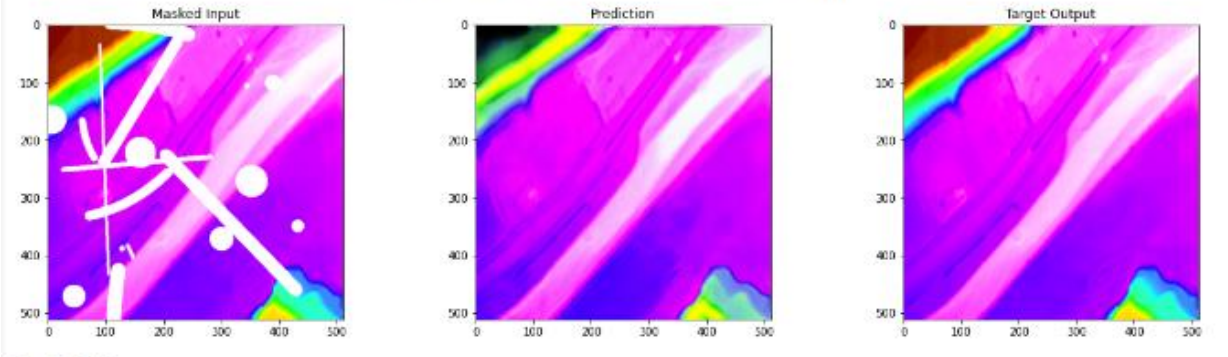
Last four epoch in CNN training



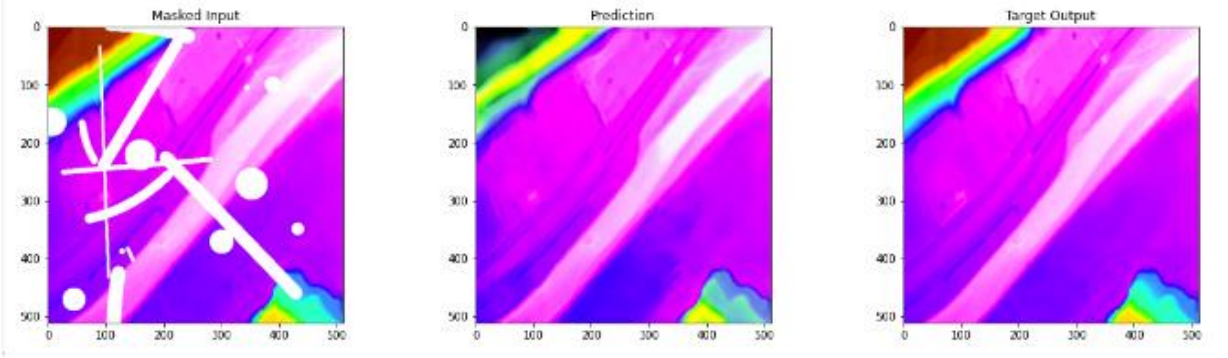

```
Epoch 8/10
1500/1500 [==============================] - 44826s 30s/step - loss: 10.4392 - PSNR: 20.3837
```




```
Epoch 9/10
1500/1500 [==============================] - 43868s 29s/step - loss: 9.6852 - PSNR: 20.7179
```




```
Epoch 10/10
1500/1500 [==============================] - 44177s 29s/step - loss: 9.1688 - PSNR: 21.0000
```



For ANN method, all the codes developed specifically for this study. All the results are testing results (out of the training zone)
Scenario a

In [1]:
```
import os
cwd=os.getcwd()
```

In [2]:
```
import pandas as pd
Gaula_data = pd.read_csv(r'C:\Users\raffaa\Desktop\ID_model\Gaula_points.csv')
#print (Gaula_data)
```

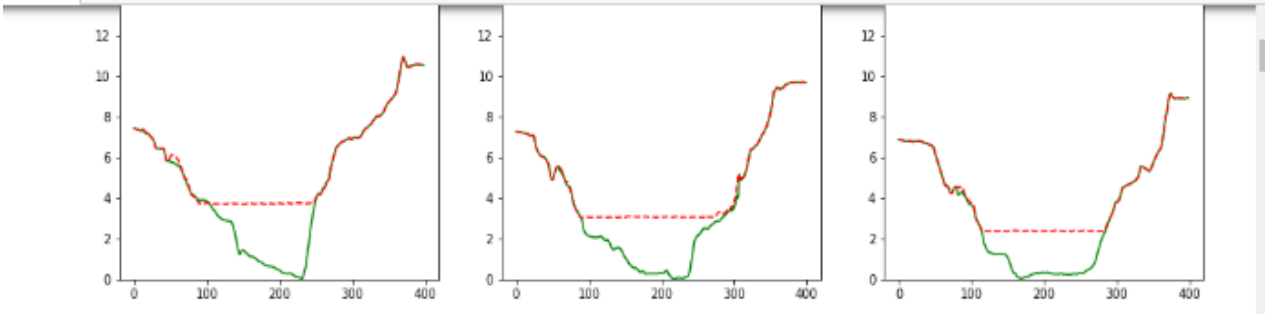
In [3]:
```
from matplotlib import pyplot as plt
print(plt.style.available)
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid', 'bmh', 'classic', 'dark_background', 'fast',
'fivethirtyeight', 'ggplot', 'grayscale', 'seaborn', 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark', 'seaborn-dark-pale
tte', 'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper', 'seaborn-pastel', 'seaborn-post
er', 'seaborn-talk', 'seaborn-ticks', 'seaborn-white', 'seaborn-whitegrid', 'tableau-colorblind10']
```

In [4]:
```
a = 101
b = 300
z = 400
#sample = random.sample()
```

In [6]:

```
from matplotlib import pyplot as plt
from matplotlib.backends.backend_pdf import PdfPages
import numpy as np
import pandas as pd

list = []
sample = np.arange(1,1000,5)
count = 1
fig = plt.figure(figsize = [16,16])
pdfFile = PdfPages("plotting.pdf")
# plt.style.use('fivethirtyeight')

for j in sample:
    data_cross = Gaula_data.loc[Gaula_data["ORIG_FID"] == j]
    data_cross = data_cross.reset_index(drop=True)
    Green = data_cross["green_elevation"]
    #Green[(a-1):b] = None
    Min = data_cross['green_elevation'].min()
    data_cross["green_elevation"] = data_cross["green_elevation"] - Min
    data_cross["red_elevation"] = data_cross["red_elevation"] - Min

    #Making sure the min Length is 399
    Length = len(data_cross)
    list.append(Length)

    #plotting

    plt.subplot(3,3,count)
    plt.plot(data_cross["green_elevation"],label ="green elevation",color = "g",linestyle= "-")
    plt.plot(data_cross["red_elevation"],"--r",label ="red elevation")
    plt.legend(loc = "upper left", fontsize = 10)
    plt.title(j)
    plt.ylim([0,16])

    if count < 9:
        count += 1
    else:
        count = 1
        pdfFile.savefig(fig)
        fig = plt.figure(figsize = [16,16])


pdfFile.close()
```



Trainning

[11]:
```
import tensorflow as tf
from keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
```

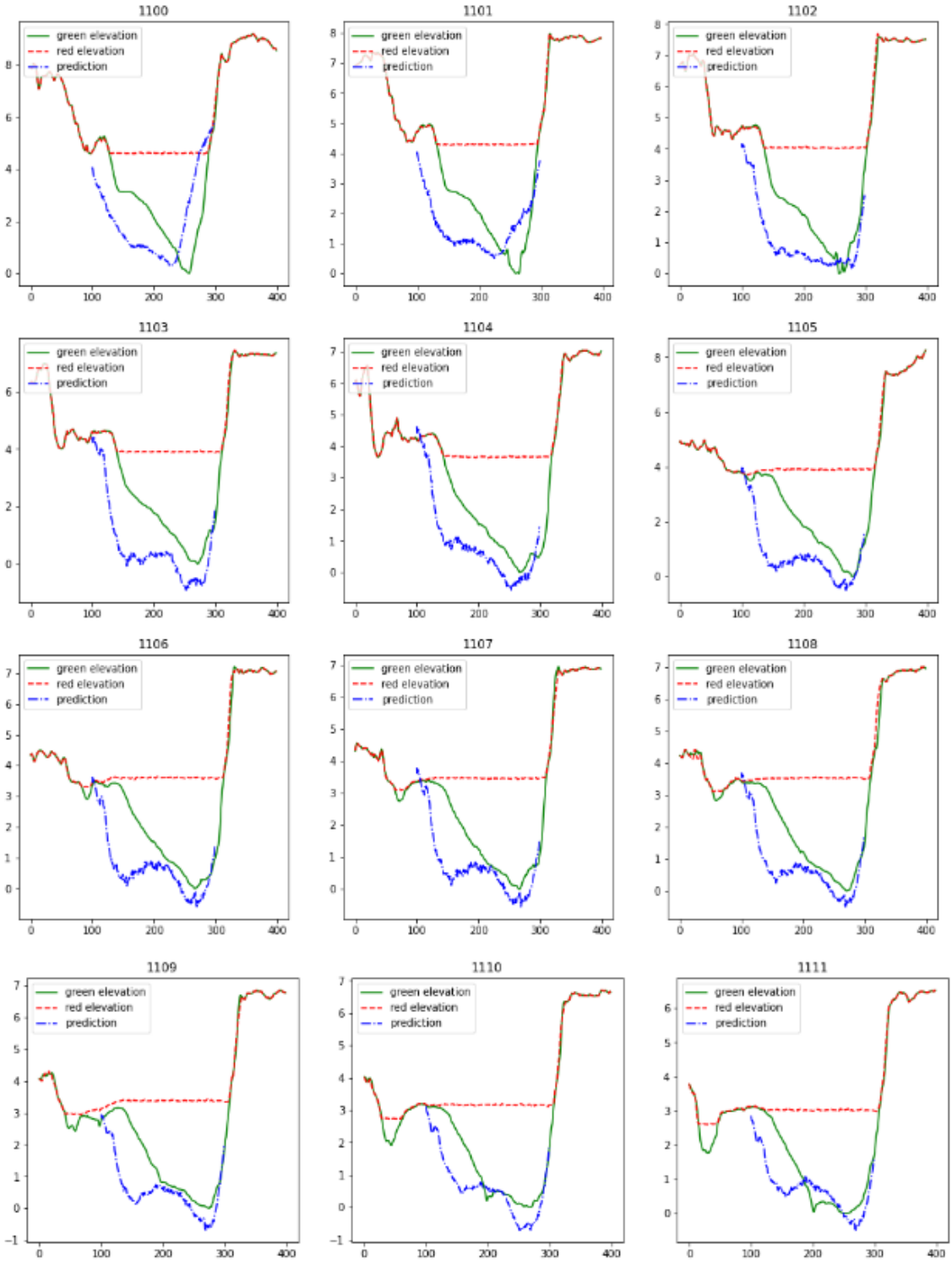
[12]:
```
# model creation & adding layers
model = Sequential([
    Dense(164, activation= "relu", input_shape=(200,)),
    Dense(164, activation= "relu"),
    Dense(200, activation= "linear"),
])

model.summary()
```

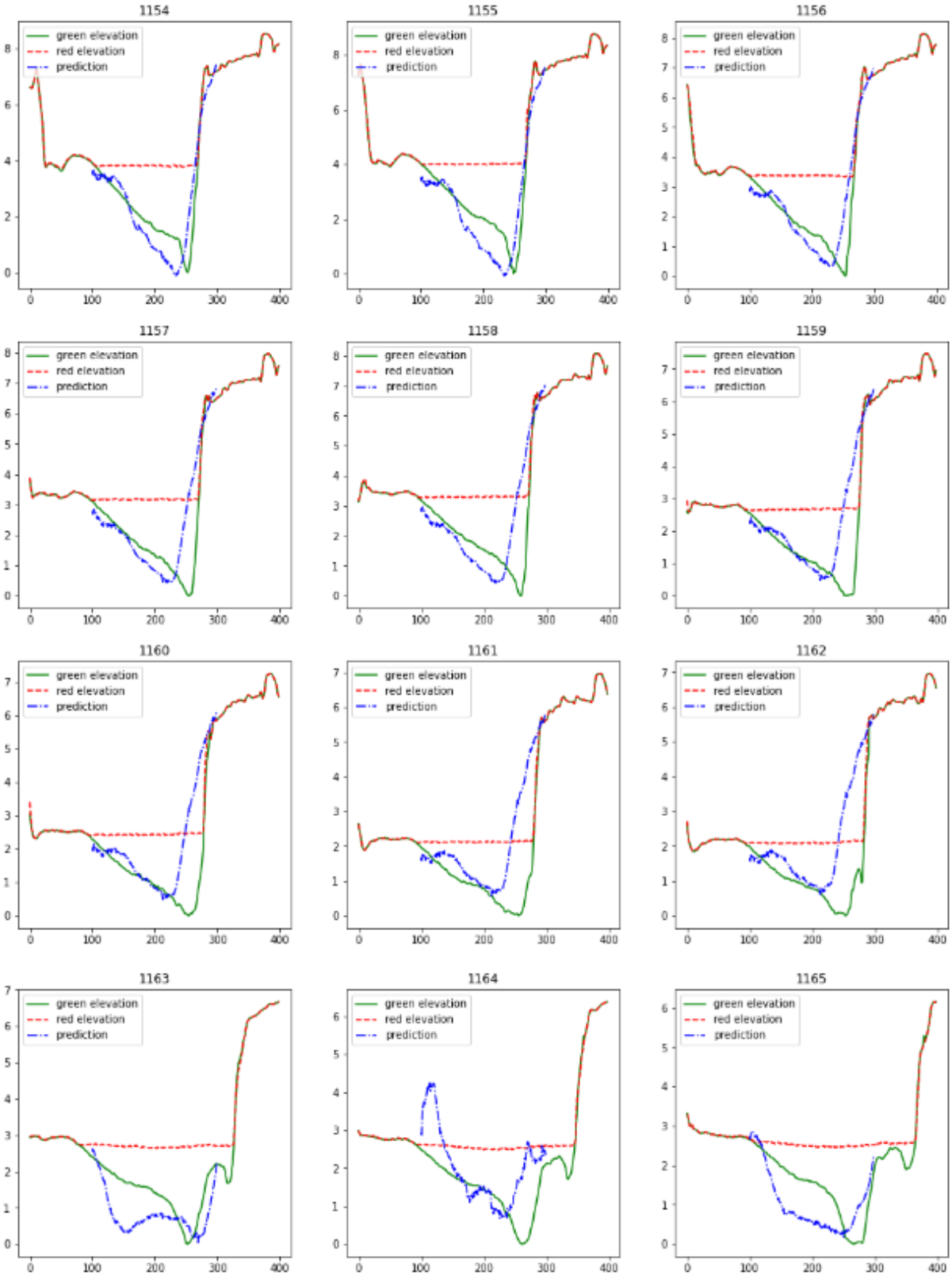
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 164)               32964

 dense_1 (Dense)             (None, 164)               27060

 dense_2 (Dense)             (None, 200)               33000

=================================================================
Total params: 93,024
Trainable params: 93,024
Non-trainable params: 0
_________________________________________________________________
```

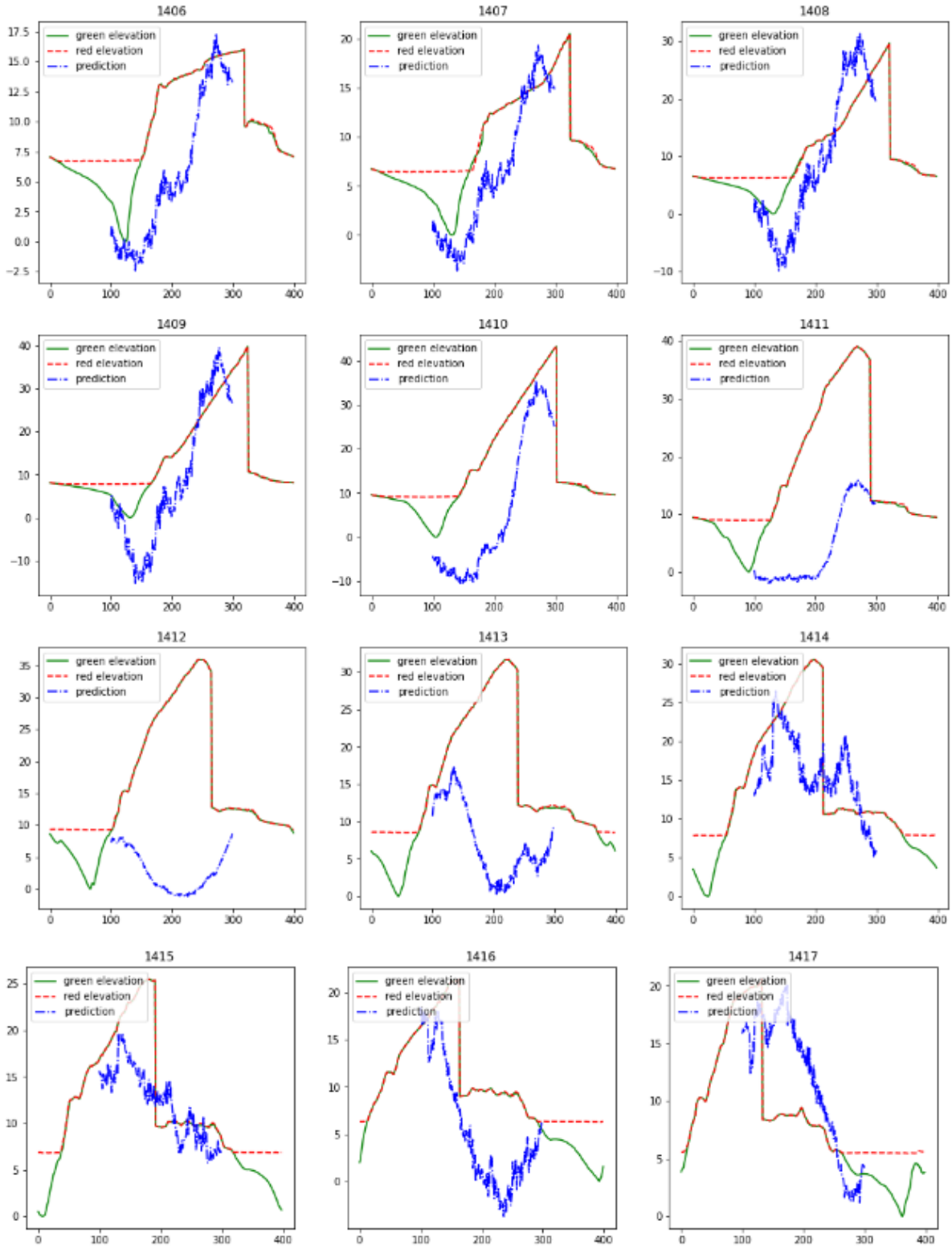
[13]:
```
# compile model
model.compile(optimizer ="rmsprop",
              loss= "mse")
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 164)               32964

 dense_1 (Dense)             (None, 164)               27060

 dense_2 (Dense)             (None, 200)               33000

=================================================================
Total params: 93,024
Trainable params: 93,024
Non-trainable params: 0
_________________________________________________________________
```

[14]:
```
# Trainning
model.fit(
input1,
Middle,
batch_size=1,
epochs=250, verbose=2,
callbacks=None,
validation_split=0.2,
validation_data=None,
shuffle=True,
class_weight=None,
sample_weight=None,
initial_epoch=0)
```


1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
green elevation
red elevation
prediction


1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
green elevation
red elevation
prediction


1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
green elevation
red elevation
prediction

Scenario b

```
import os
cwd=os.getcwd()
```

```
import numpy as np
import pandas as pd
Gaula_data = pd.read_csv(r'C:\Users\raffaa\Desktop\ID_model\Gaula_points.csv')
#print (Gaula_data)
```

#

```
middle_points = pd.read_csv(r'C:\Users\raffaa\Desktop\ID_model\Points_clip.csv')
```

```
middle_points= middle_points.sort_values(by=["copy_id"],ascending=True)
```

```
middle_points
```

```
Middle1.iat[0,0] = 5
```

```
Middle1
```

```
import pandas as pd

Middle1 = pd.DataFrame(0, index= np.arange(0,2000,1), columns = [np.arange(0,863,1)])

sample = np.arange(0,2000,1)
list =[]
count = 0
for j in sample:
    data_cross = middle_points.loc[middle_points["transectID"] == j]
    data_cross = data_cross.reset_index(drop=True)
    Min = data_cross['Green_Lida'].min()
    data_cross["Green_Lida"] = data_cross["Green_Lida"] - Min
    data_cross["Red_Lidar"] = data_cross["Red_Lidar"] - Min
    Green = data_cross["Green_Lida"]

    list.append(len(Green))

    Middle1.iloc[count,0:len(Green)] = Green.values
    #print(Len(Green))
    count= count +1
#max(List)
```

```
Middle1
```

```
np.shape(Middle1)
```

##

```
Banks_points = pd.read_csv(r'C:\Users\raffaa\Desktop\ID_model\main_points.csv')
```

```
Banks_points= Banks_points.sort_values(by=["copy_id"],ascending=True)
```

```
Banks_points
```

```
import pandas as pd


left_right = pd.DataFrame(0, index= np.arange(0,2000,1), columns = [np.arange(0,863,1)])

sample = np.arange(1,2000,1)
list =[]
count = 0
for j in sample:
    data_cross = Banks_points.loc[Banks_points["transectID"] == j]
    data_cross = data_cross.reset_index(drop=True)
    Min = data_cross['Green_Lida'].min()
    data_cross["Green_Lida"] = data_cross["Green_Lida"] - Min
    data_cross["Red_Lidar"] = data_cross["Red_Lidar"] - Min
    Green = data_cross["Green_Lida"]

    list.append(len(Green))

    left_right.iloc[count,0:len(Green)] = Green.values
    #print(Len(Green))
    count= count +1
#max(List)
min(list)
```

```
left_right
```

```
np.shape(left_right)
```

##

```
!pip install keras==2.6
```

Training

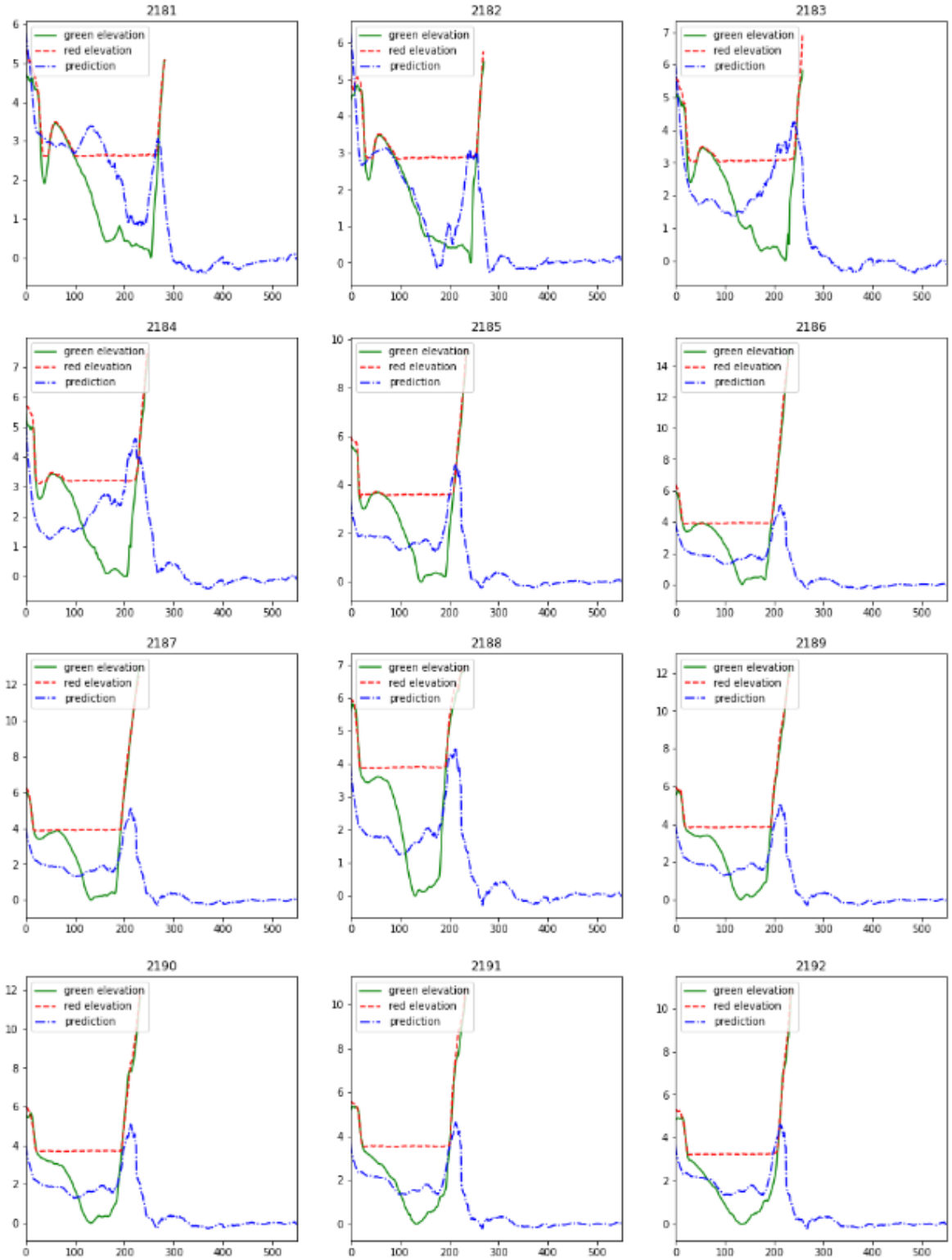
```
import tensorflow as tf
from keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
```

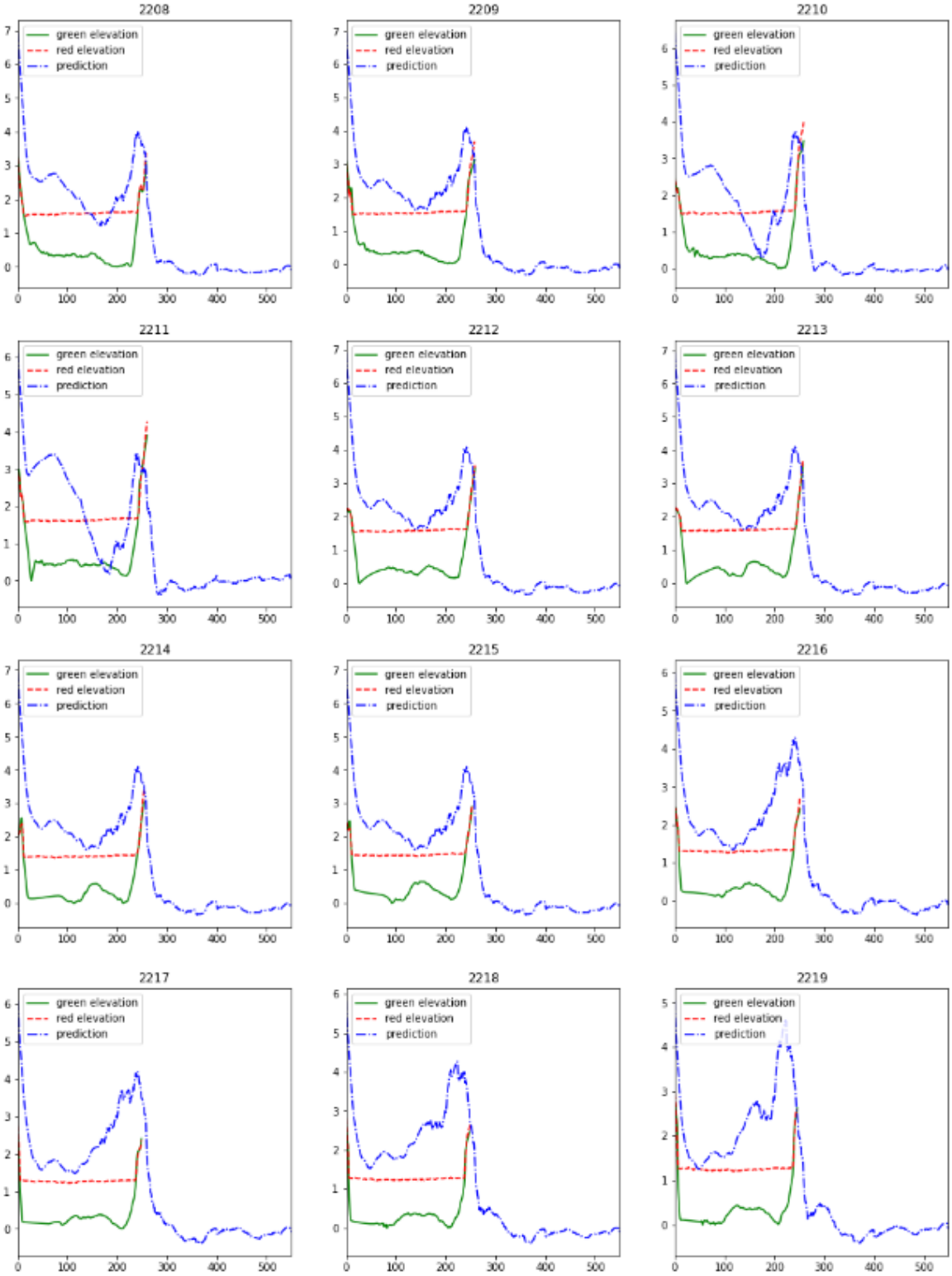
```
# model creation & adding layers
model = Sequential([
    Dense(164, activation= "sigmoid", input_shape=(863,)),
    Dense(164, activation= "sigmoid"),
    Dense(863, activation= "linear"),
])

model.summary()
```

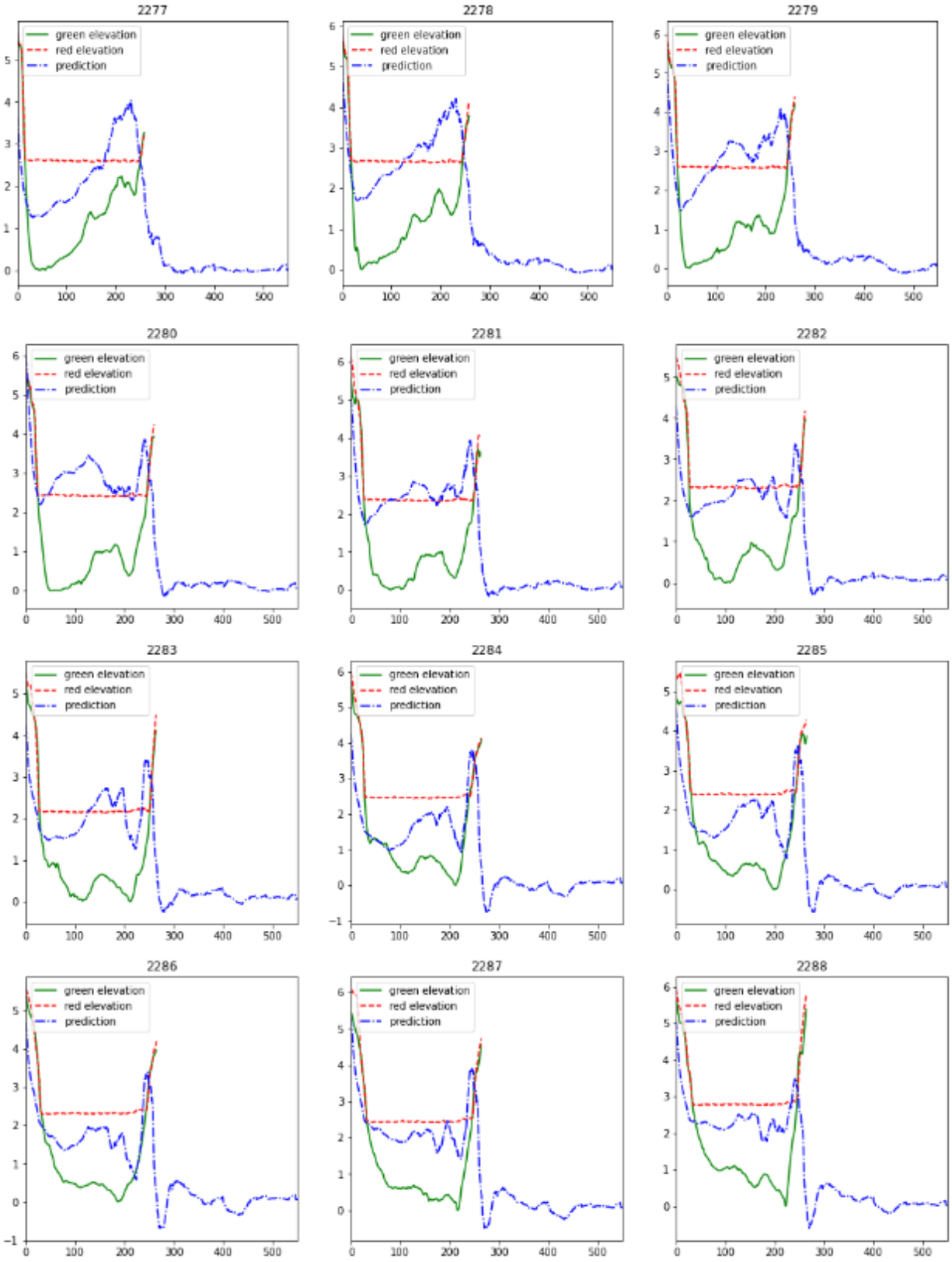
```
# compile model
model.compile(optimizer ="rmsprop",
              loss= "mse")
model.summary()
```

```
# Trainning
model.fit(
left_right,
Middle1,
batch_size=1,
epochs=200, verbose=2,
callbacks=None,
validation_split=0.2,
validation_data=None,
shuffle=True,
class_weight=None,
sample_weight=None,
initial_epoch=0)
```

```
]: import numpy as np
   import pandas as pd
   from matplotlib.backends.backend_pdf import PdfPages
   from matplotlib import pyplot as plt

   count = 1

   fig = plt.figure(figsize = [16,16])
   pdfFile = PdfPages("Zeros_predictions.pdf")

   for j in range(2100,2400):
       data_cross = middle_points.loc[middle_points["transectID"] == j]
       data_cross = data_cross.reset_index(drop=True)

       Min = data_cross['Green_Lida'].min()
       data_cross["Green_Lida"] = data_cross["Green_Lida"] - Min
       data_cross["Red_Lidar"] = data_cross["Red_Lidar"] - Min
       Green = data_cross["Green_Lida"]


       #filling the df with the prdiction input for j cross section

       df = pd.DataFrame(0, index= np.arange(0,1,1), columns = [np.arange(0,863,1)])

       data_predict = Banks_points.loc[Banks_points["transectID"] == j]
       data_predict = data_predict.reset_index(drop=True)
       Min = data_predict['Green_Lida'].min()
       data_predict["Green_Lida"] = data_predict["Green_Lida"] - Min
       data_predict["Red_Lidar"] = data_predict["Red_Lidar"] - Min
       Green_predict = data_predict["Green_Lida"]

       df.iloc[0,0:len(Green_predict)] = Green_predict.values




       #input_pr = pd.merge(Left,Right,left_index=True, right_index=True)
       predictions = model.predict(df,batch_size=1,verbose=1)
       predictions = np.transpose(predictions)

       #print(input_pr)
       pred = pd.DataFrame(predictions)
       ind1 = np.arange(0,863,1)

       plt.subplot(3,3,count)
       plt.plot(data_cross["Green_Lida"],label ="green elevation",color = "g",linestyle= "-")
       plt.plot(data_cross["Red_Lidar"],"--r",label ="red elevation")
       plt.plot(ind1,pred[0],label ="prediction",color = "b",linestyle= "-.")
       plt.legend(loc = "upper left", fontsize = 10)
       plt.title(j)
       #plt.ylim([0,16])
       plt.xlim([0,550])
       if count < 9:
           count += 1
       else:
           count = 1
           pdfFile.savefig(fig)
           fig = plt.figure(figsize = [16,16])


   pdfFile.close()
```


2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
green elevation
red elevation
prediction


2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
green elevation
red elevation
prediction


2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
green elevation
red elevation
prediction

Scenario c

In [1]:
```
import pandas as pd
Gaula_data = pd.read_csv(r'C:\Users\raffaa\Desktop\ID_model\Gaula_points.csv')
```

In [2]:
```
Gaula_data
```

Out[2]:

	OID_	ORIG_FID	ORIG_FID_1	red_elevation	green_elevation	POINT_X	POINT_Y
0	1	1	200	33.745838	33.745838	562748.6648	7.001246e+06
1	2	1	200	33.736206	33.736206	562748.1769	7.001246e+06
2	3	1	200	33.730690	33.730690	562747.6889	7.001246e+06
3	4	1	200	33.711281	33.711281	562747.2010	7.001246e+06
4	5	1	200	33.695003	33.695003	562746.7130	7.001246e+06
...	...	...	...	...	...	...	...
1045880	1045881	2618	200	13.830121	13.830121	564226.5692	7.020697e+06
1045881	1045882	2618	200	13.754012	13.754012	564227.0542	7.020697e+06
1045882	1045883	2618	200	13.770851	13.770851	564227.5392	7.020697e+06
1045883	1045884	2618	200	13.831068	13.831068	564228.0242	7.020697e+06
1045884	1045885	2618	200	13.825141	13.825141	564228.5092	7.020698e+06

1045885 rows × 7 columns

In [3]:
```
z= 400 #change
```

In [4]:
```
import numpy as np
import pandas as pd

Red_frame = pd.DataFrame(0, index= np.arange(0,2300,1), columns = [np.arange(0,400,1)]) #change

sample = np.arange(1,2301,1)
list =[]
count = 0
for j in sample:
    data_cross = Gaula_data.loc[Gaula_data["ORIG_FID"] == j]
    data_cross = data_cross.reset_index(drop=True)
    Min = data_cross['green_elevation'].min()
    data_cross["green_elevation"] = data_cross["green_elevation"] - Min
    data_cross["red_elevation"] = data_cross["red_elevation"] - Min
    Red = data_cross["red_elevation"]

    list.append(len(Red))

    if len(Red)== z-1:
        new_row = Red.iloc[z-2]
        Red.loc[z-1] = new_row

    Red_frame.iloc[count,0:len(Red)] = Red.values
    #print(len(Red))
    count= count +1
#print(max(list))
#print(min(list))
```

In [6]:
```
import numpy as np
import pandas as pd

Green_frame = pd.DataFrame(0, index= np.arange(0,2300,1), columns = [np.arange(0,400,1)]) #change

sample = np.arange(1,2301,1)
list =[]
count = 0
for j in sample:
    data_cross = Gaula_data.loc[Gaula_data["ORIG_FID"] == j]
    data_cross = data_cross.reset_index(drop=True)
    Min = data_cross['green_elevation'].min()
    data_cross["green_elevation"] = data_cross["green_elevation"] - Min
    data_cross["red_elevation"] = data_cross["red_elevation"] - Min
    Green = data_cross["green_elevation"]

    list.append(len(Green))

    if len(Green)== z-1:
        new_row = Green.iloc[z-2]
        Green.loc[z-1] = new_row

    Green_frame.iloc[count,0:len(Green)] = Green.values
    #print(len(Red))
    count= count +1
#print(max(list))
#print(min(list))
```

In [8]:
```
import tensorflow as tf
from keras.models import Sequential
from tensorflow.keras.layers import Activation, Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.metrics import categorical_crossentropy
```

In [9]:
```
# model creation & adding layers
model = Sequential([
    Dense(164, activation= "sigmoid", input_shape=(400,)),
    Dense(164, activation= "sigmoid"),
    Dense(400, activation= "linear"),
])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 164)               65764

 dense_1 (Dense)             (None, 164)               27060

 dense_2 (Dense)             (None, 400)               66000

=================================================================
Total params: 158,824
Trainable params: 158,824
Non-trainable params: 0
_________________________________________________________________
```

In [10]:
```
# compile model
model.compile(optimizer ="rmsprop",
              loss= "mse")
model.summary()
```

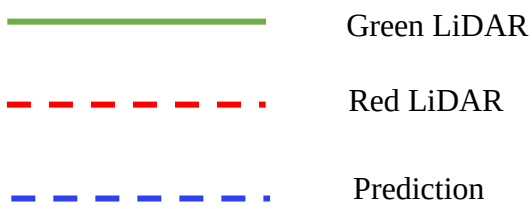
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 164)               65764

 dense_1 (Dense)             (None, 164)               27060

 dense_2 (Dense)             (None, 400)               66000

=================================================================
Total params: 158,824
Trainable params: 158,824
Non-trainable params: 0
_________________________________________________________________
```

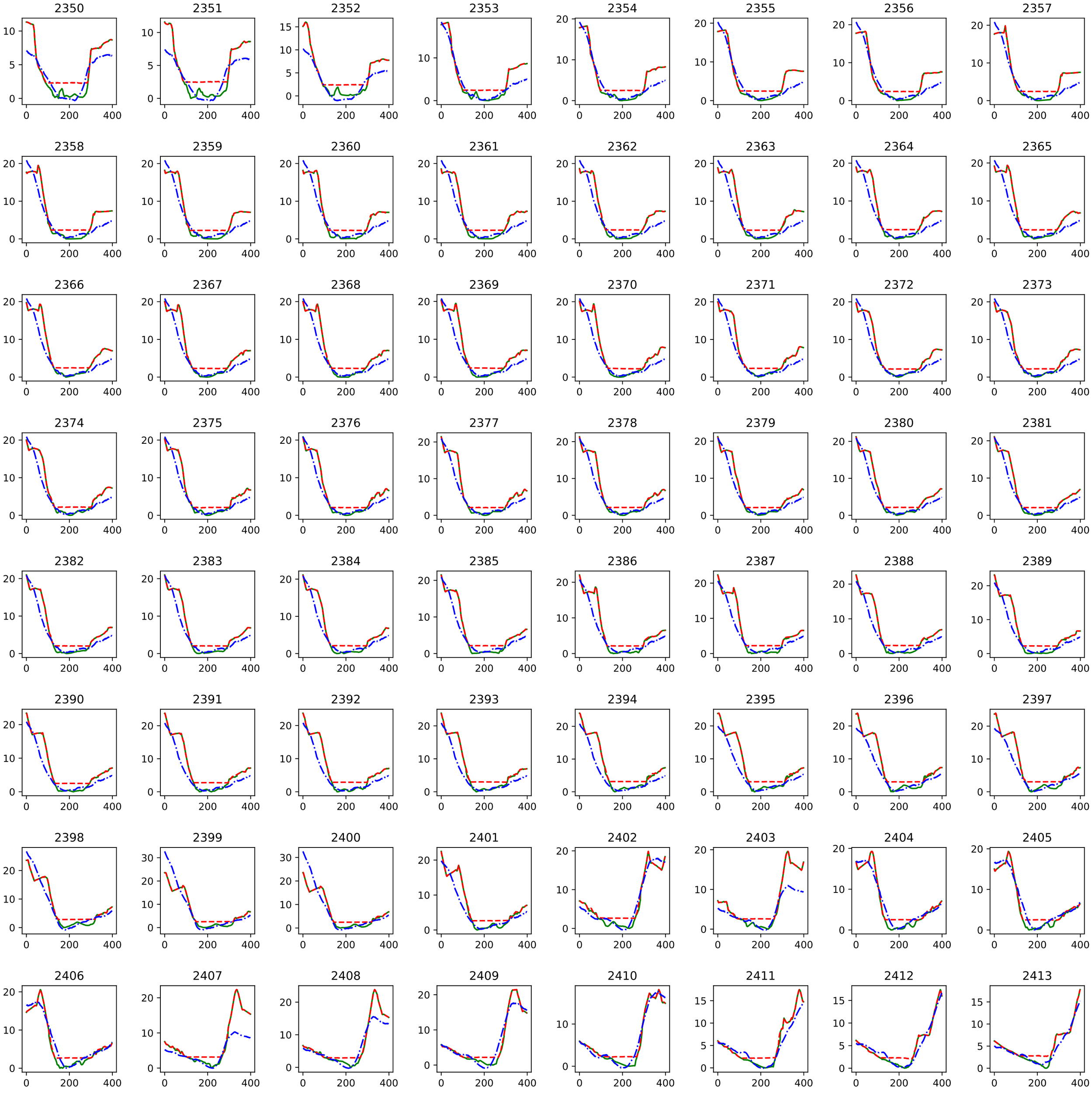
In [11]:
```
# Trainning
history=model.fit(
Red_frame,
Green_frame,
batch_size=1,
epochs=250, verbose=2,
callbacks=None,
validation_split=0.2,
validation_data=None,
shuffle=True,
class_weight=None,
sample_weight=None,
initial_epoch=0)
```

Gaula Sigmoid

In the next pages, all the cross sections predicted for Gaula outside the range of traning are shown using Sigmoid activation function.

The legend for all the graphs as follow:




2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413

2414 2415 2416 2417 2418 2419 2420 2421

2422 2423 2424 2425 2426 2427 2428 2429

2430 2431 2432 2433 2434 2435 2436 2437

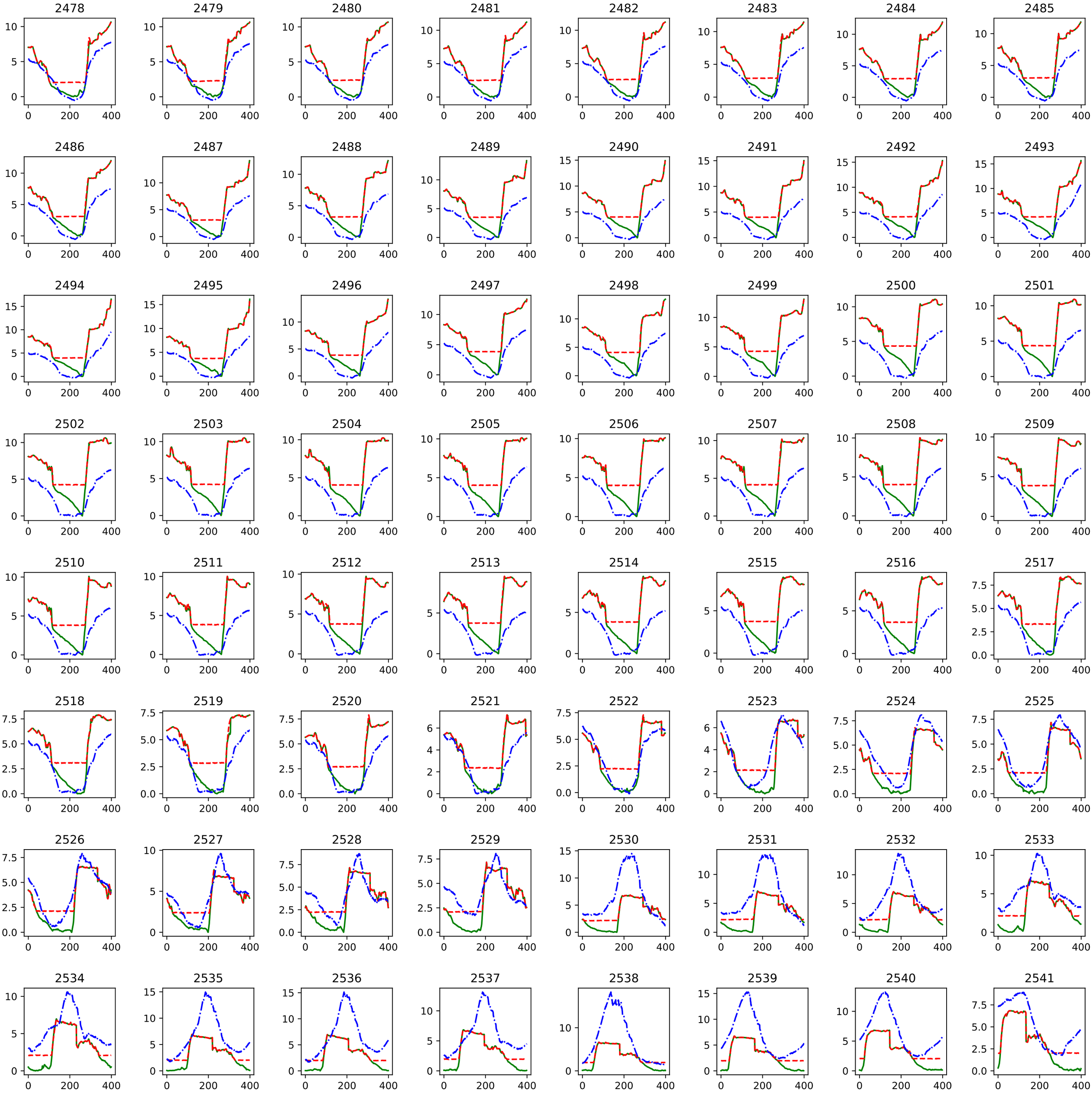
2438 2439 2440 2441 2442 2443 2444 2445

2446 2447 2448 2449 2450 2451 2452 2453

2454 2455 2456 2457 2458 2459 2460 2461

2462 2463 2464 2465 2466 2467 2468 2469

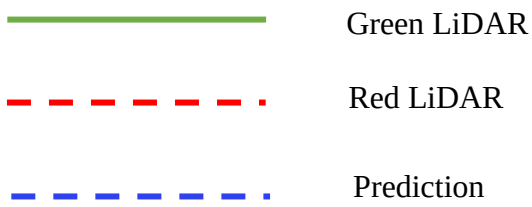
2470 2471 2472 2473 2474 2475 2476 2477

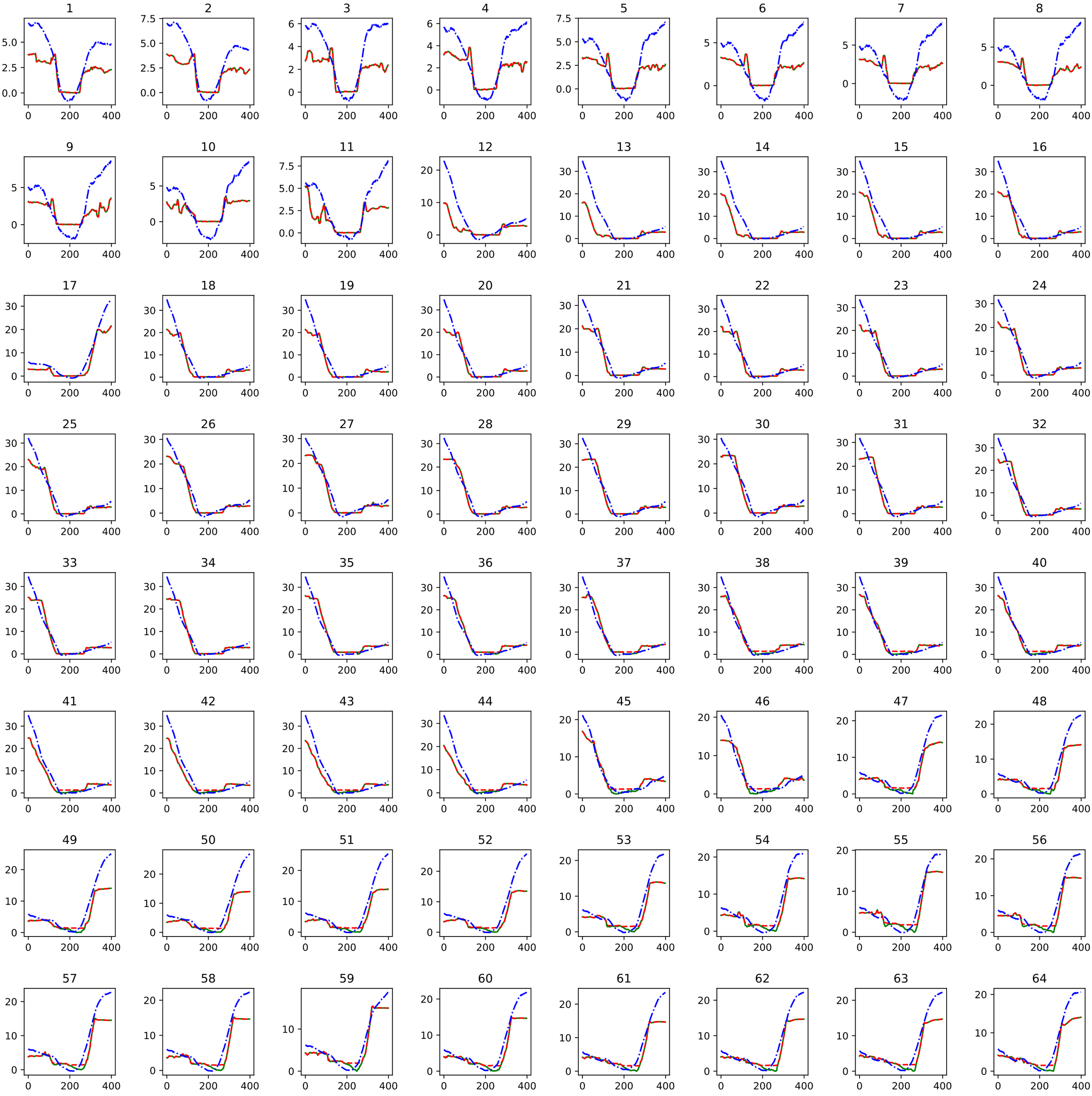

2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541

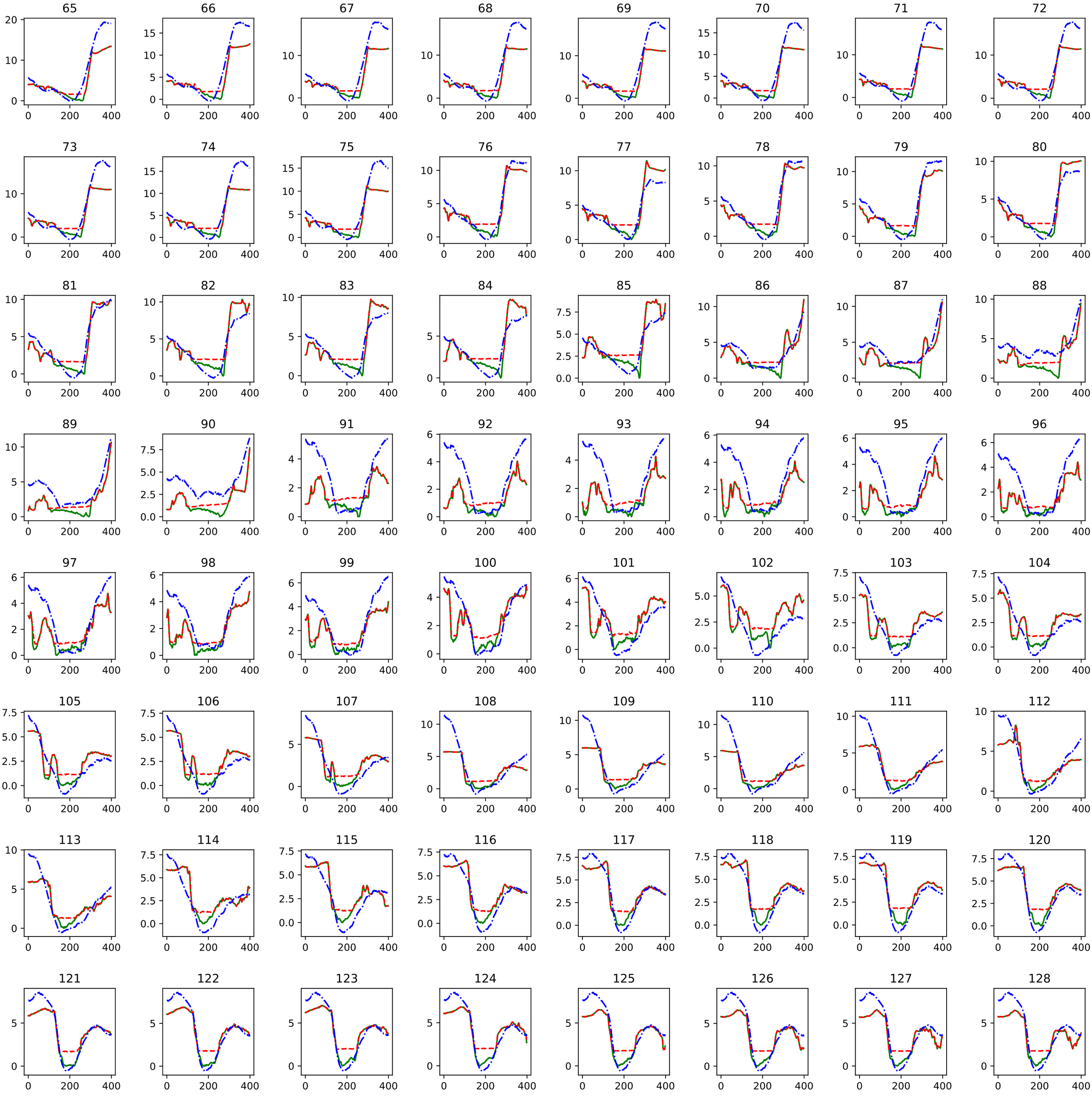
Driva Sigmoid

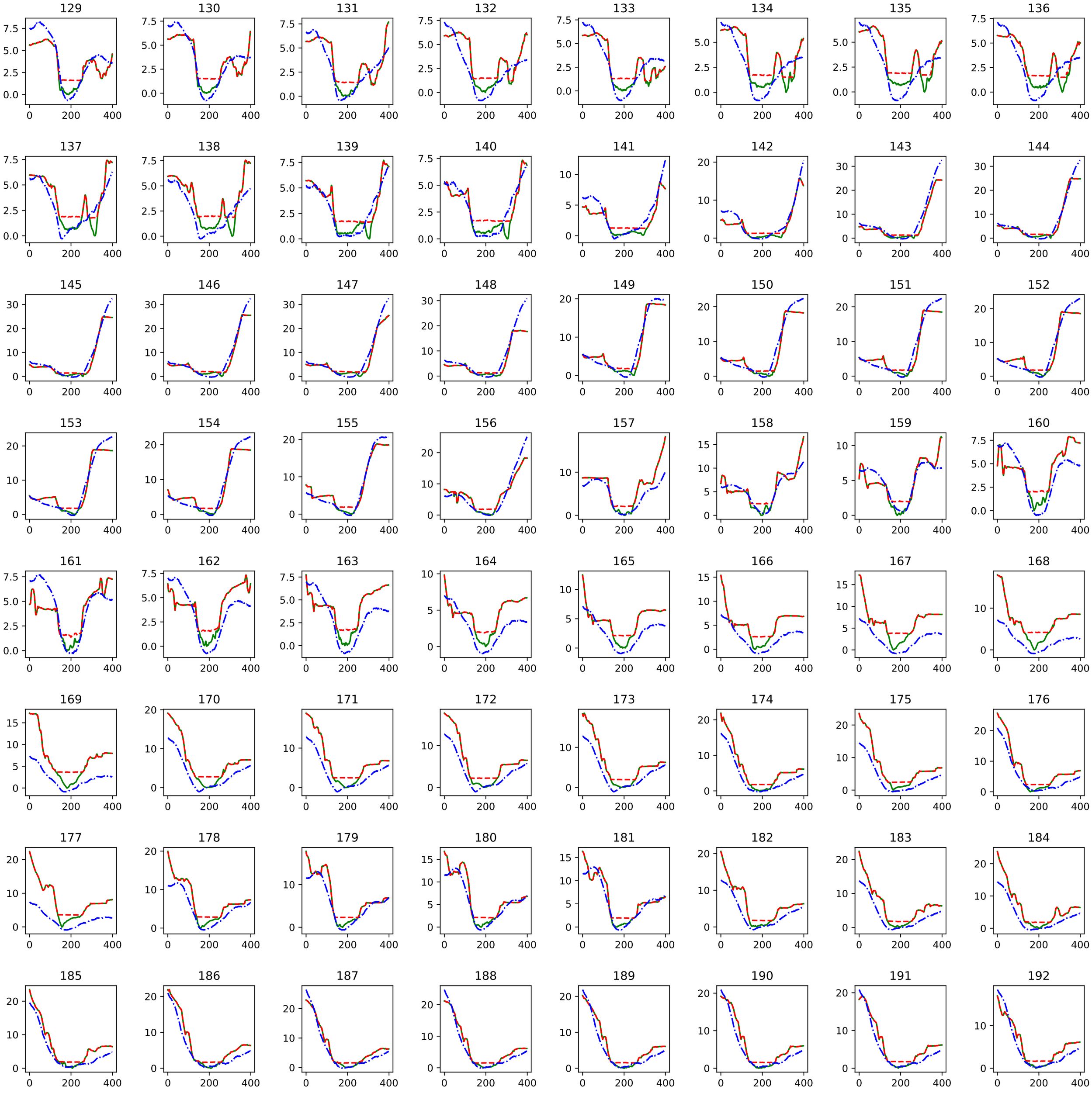
In the next pages, all the cross sections predicted for Driva are shown using Sigmoid activation function.

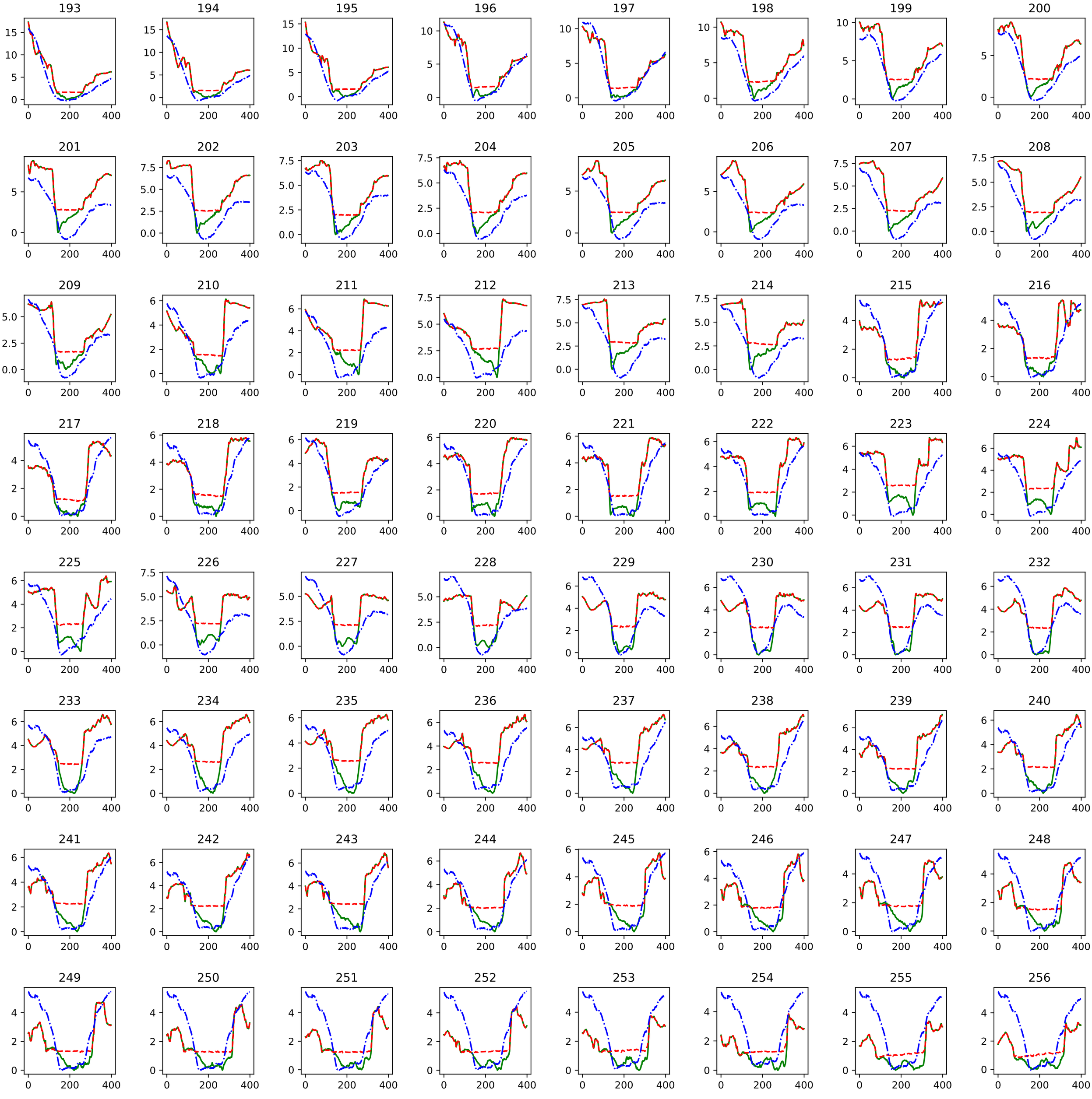
The legend for all the graphs as follow:

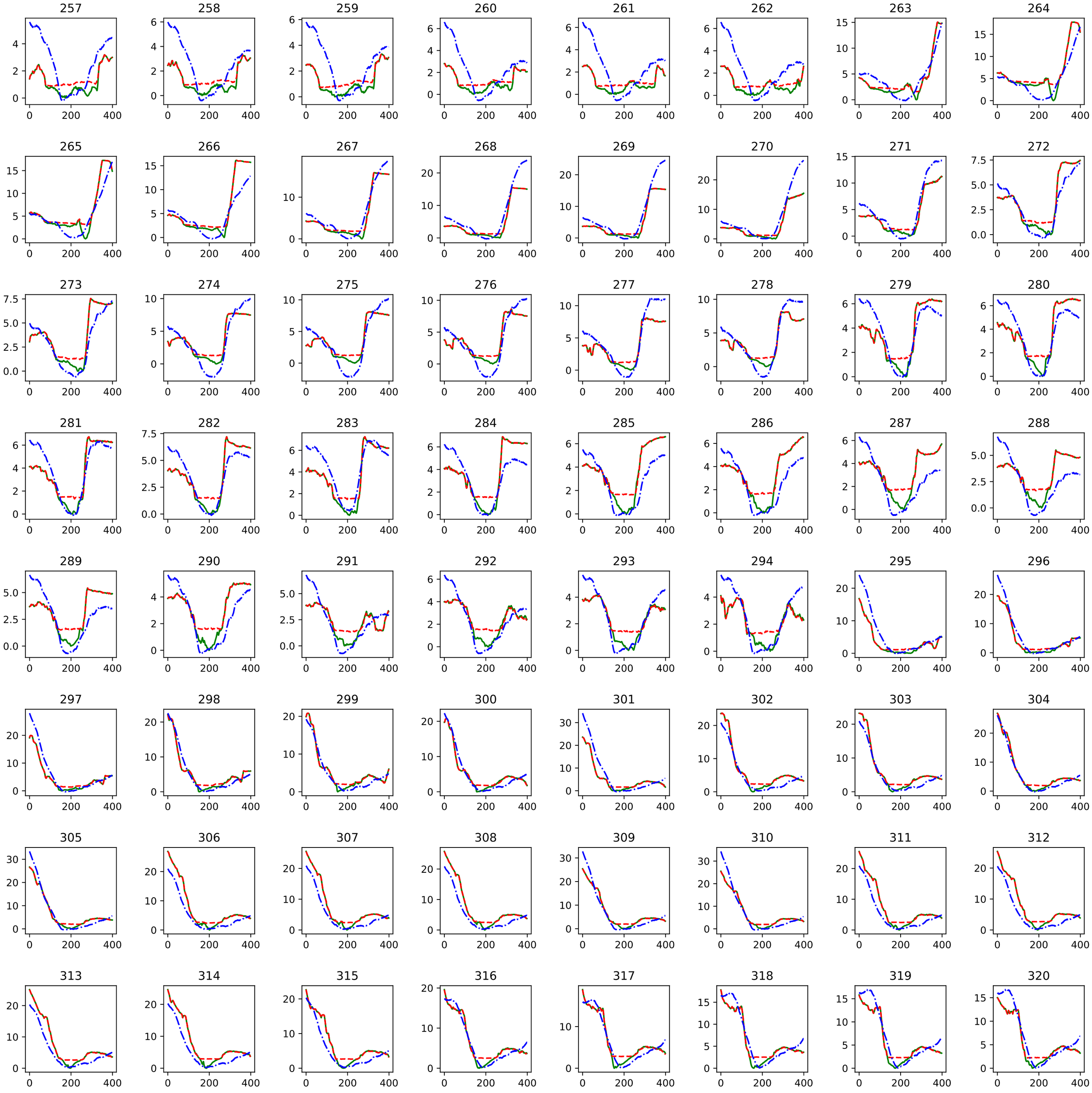


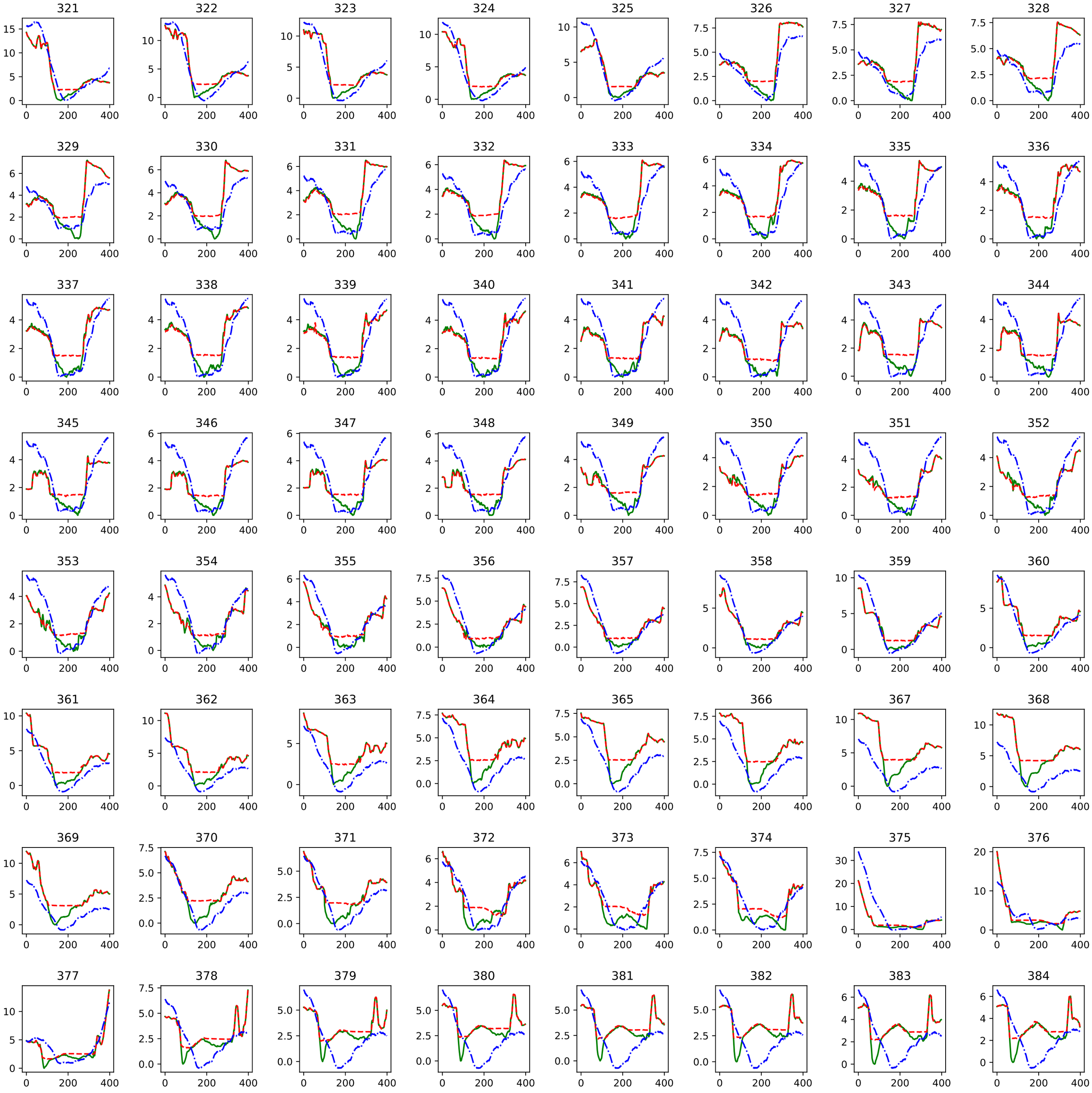

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

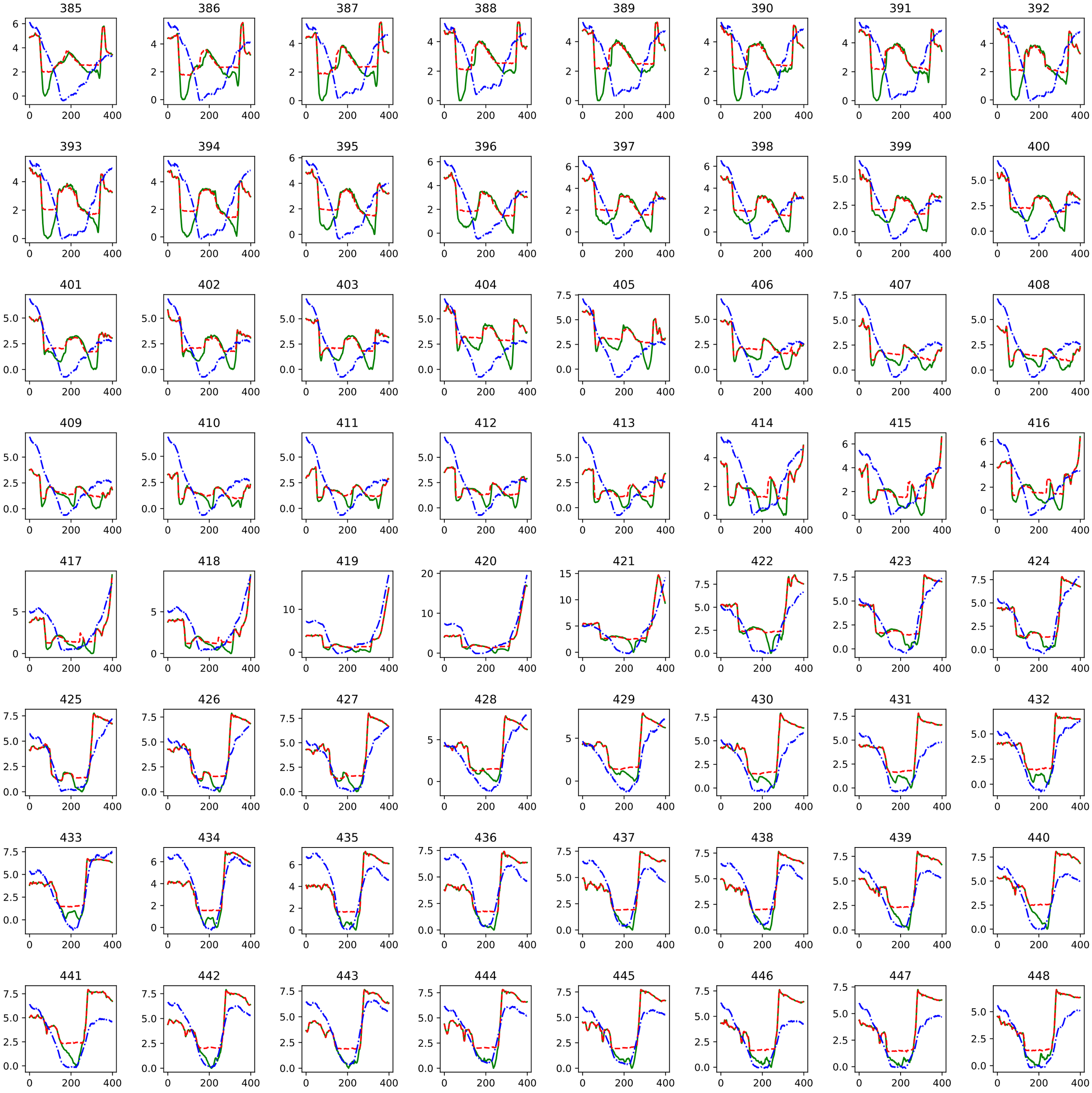

65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128

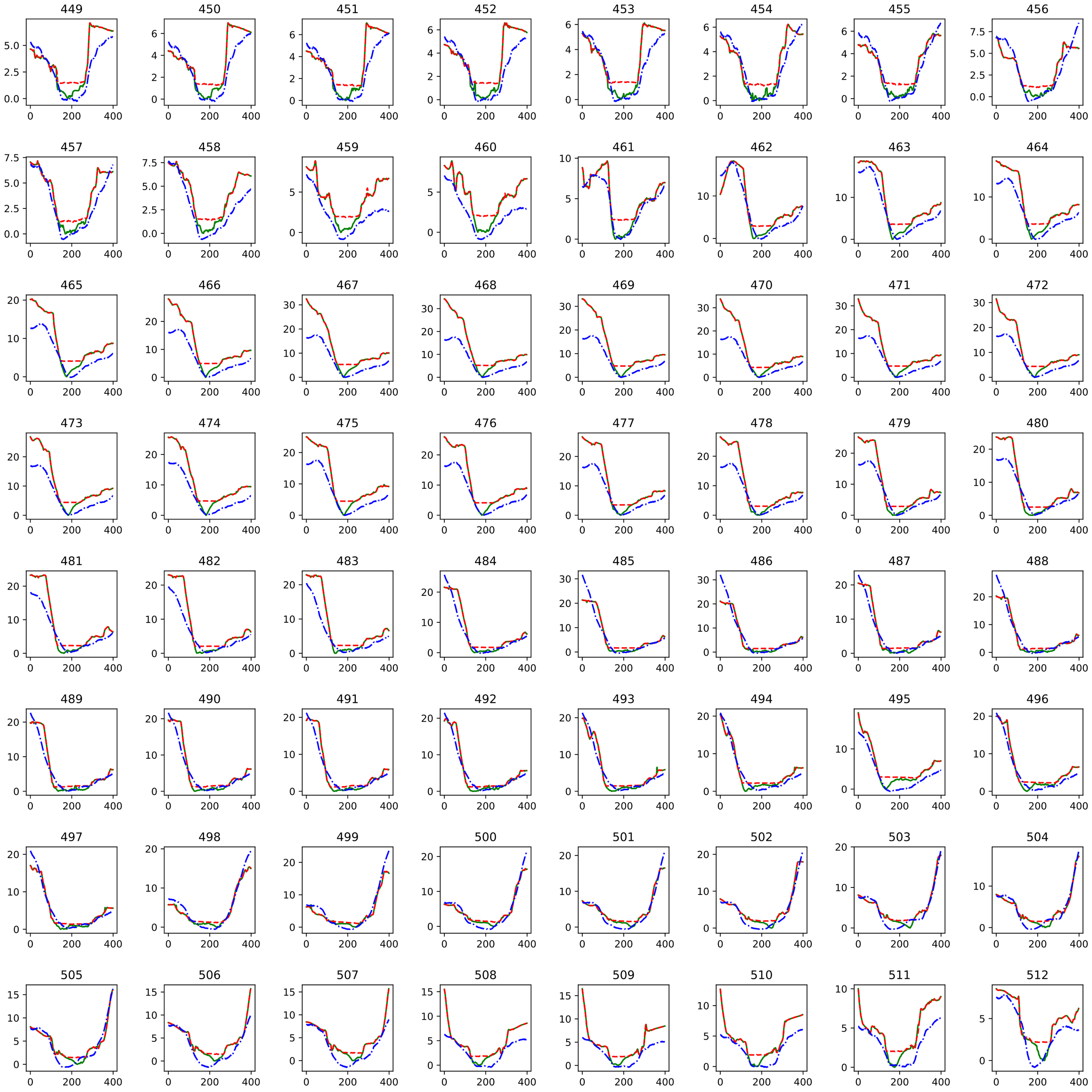

129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192


193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256


257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320


321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384


385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448


449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512

513 514 515 516 517 518 519 520

521 522 523 524 525 526 527 528

529 530 531 532 533 534 535 536

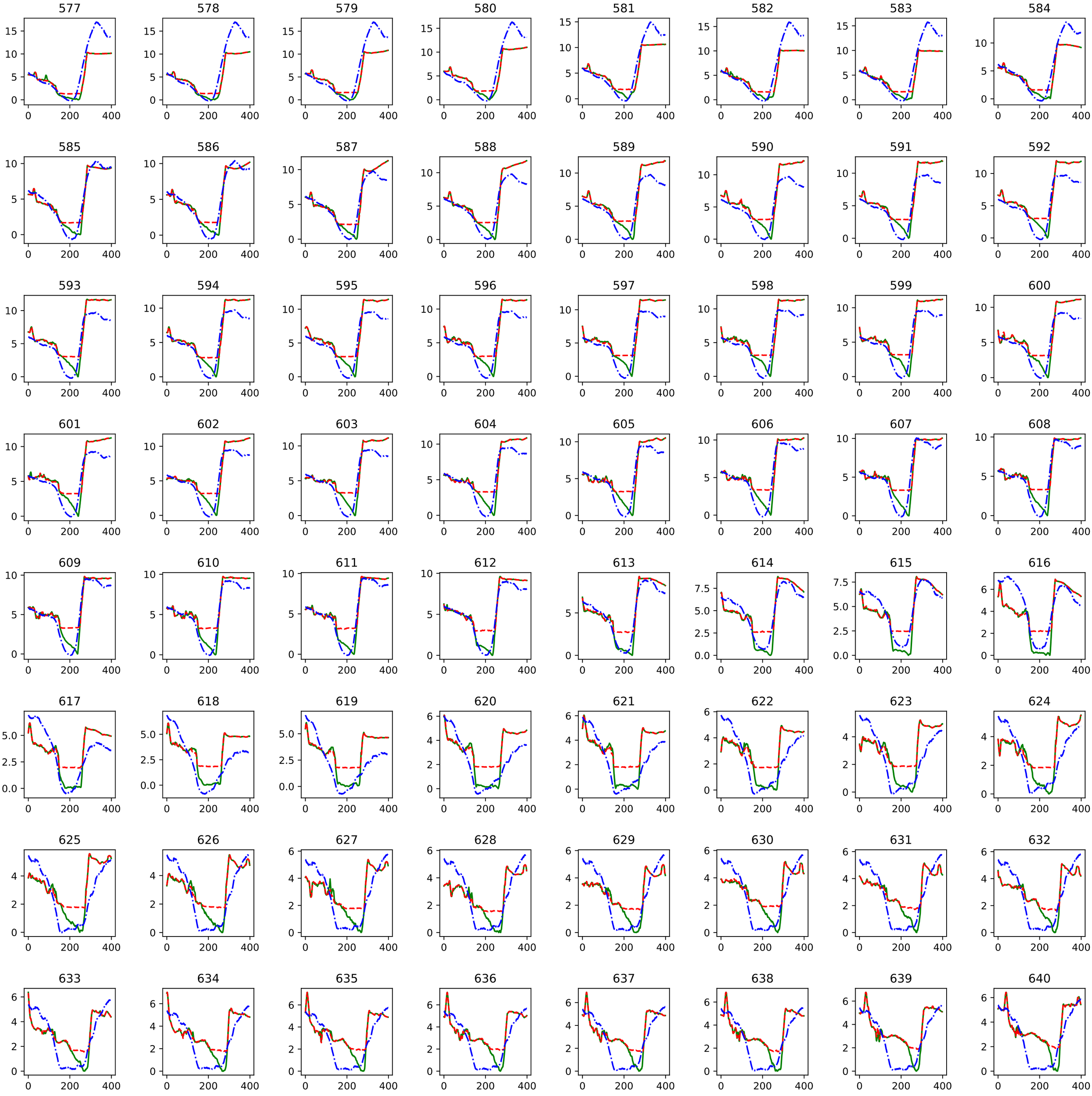
537 538 539 540 541 542 543 544

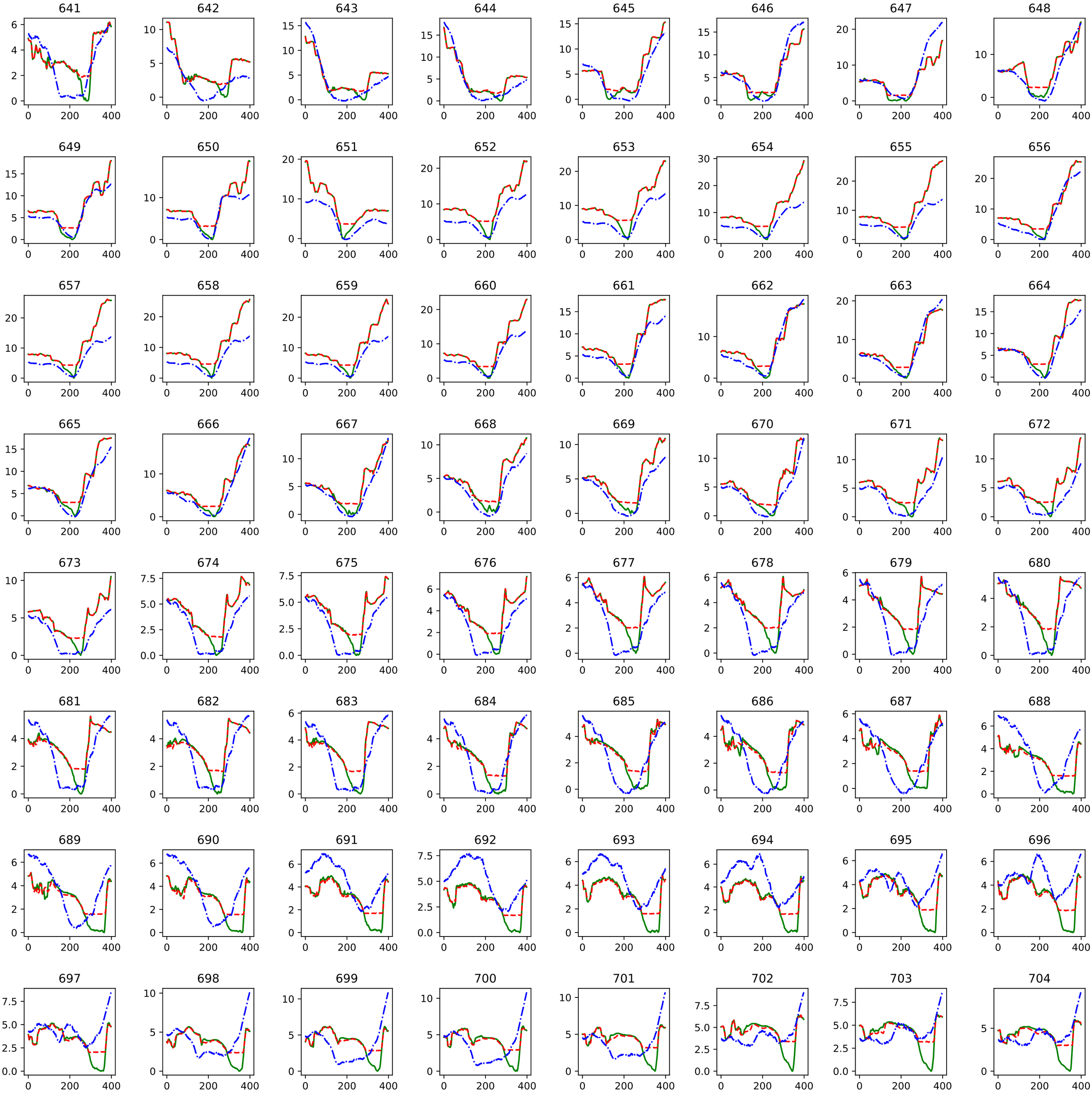
545 546 547 548 549 550 551 552

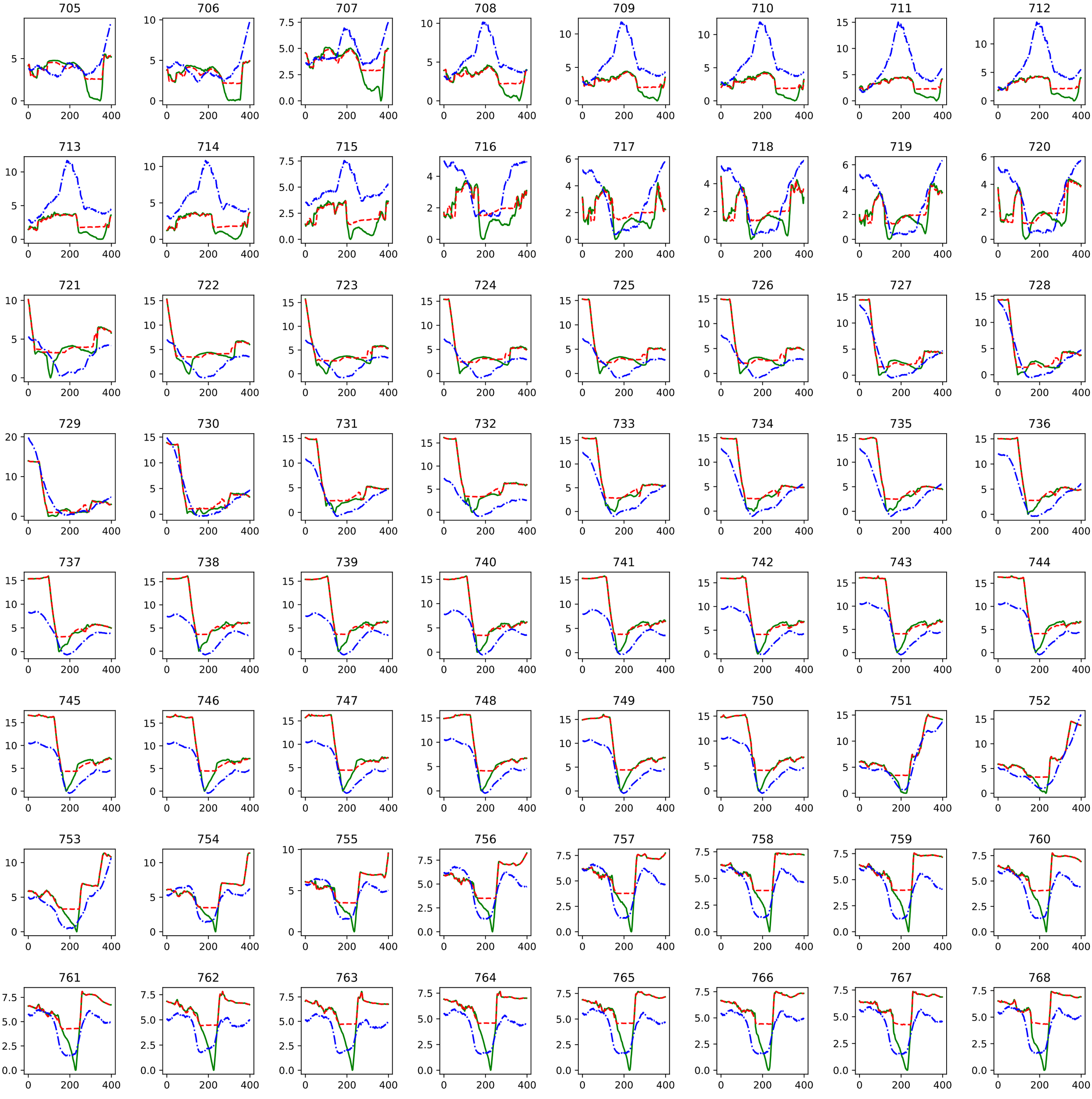
553 554 555 556 557 558 559 560

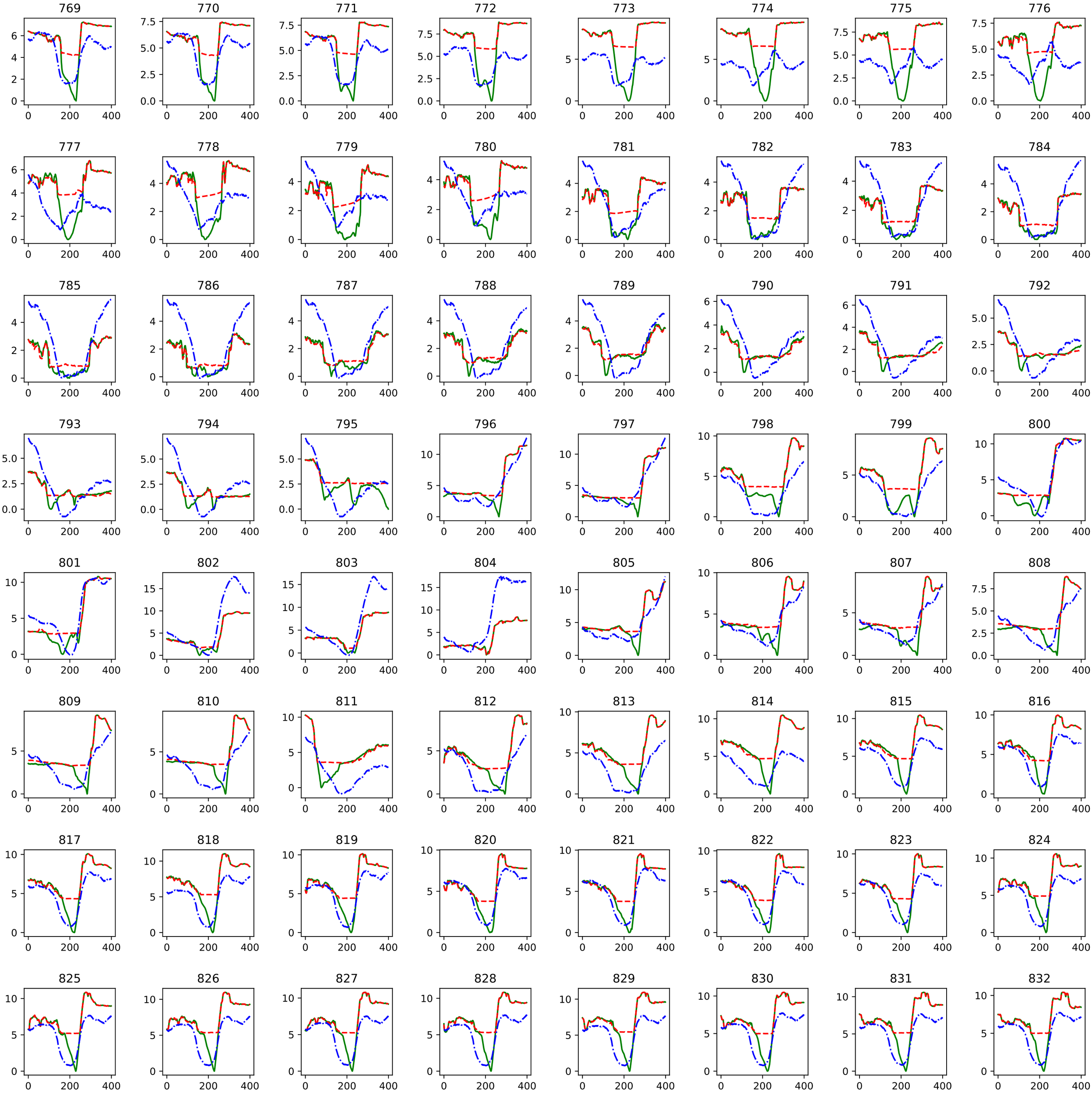
561 562 563 564 565 566 567 568

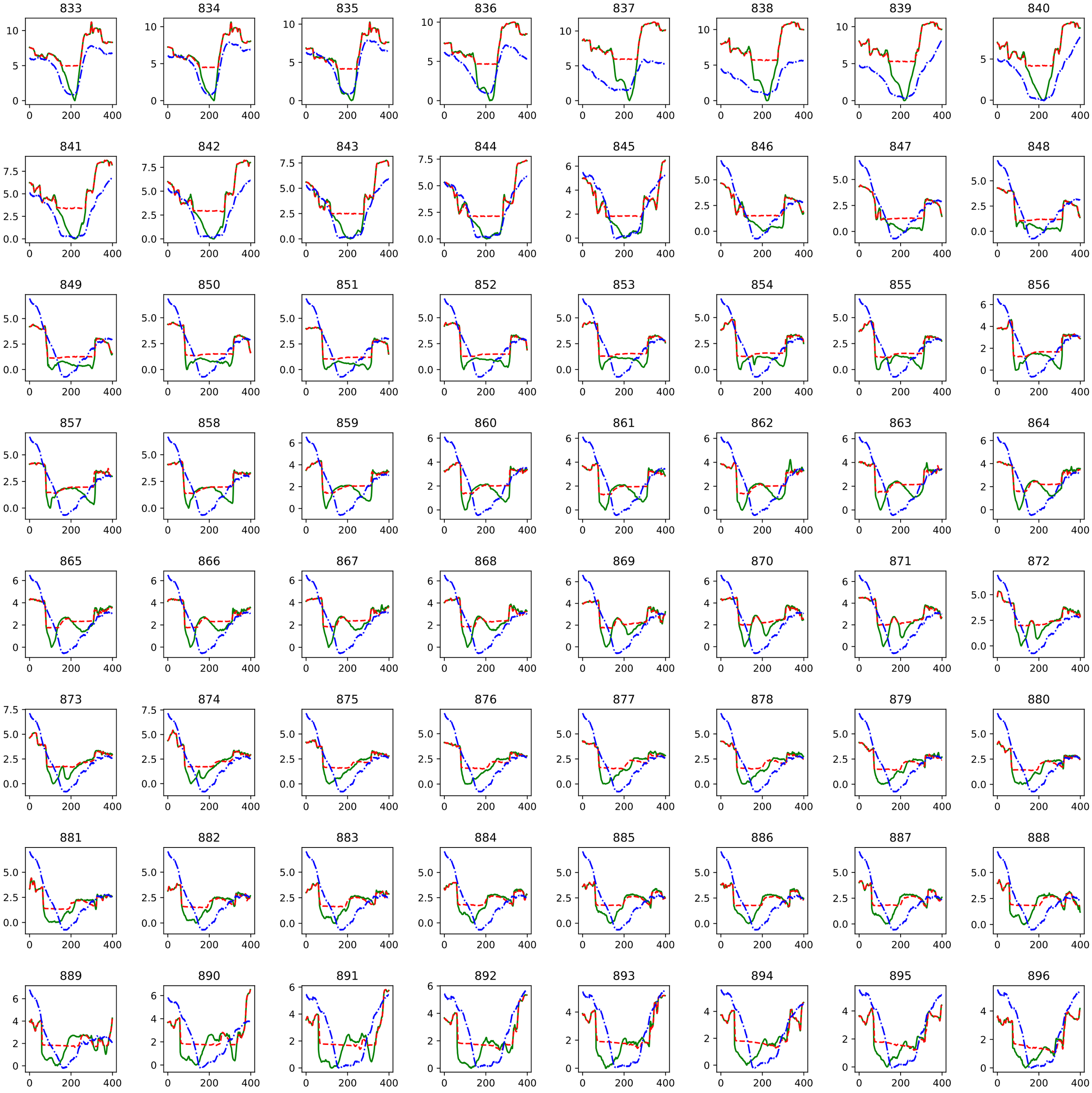
569 570 571 572 573 574 575 576

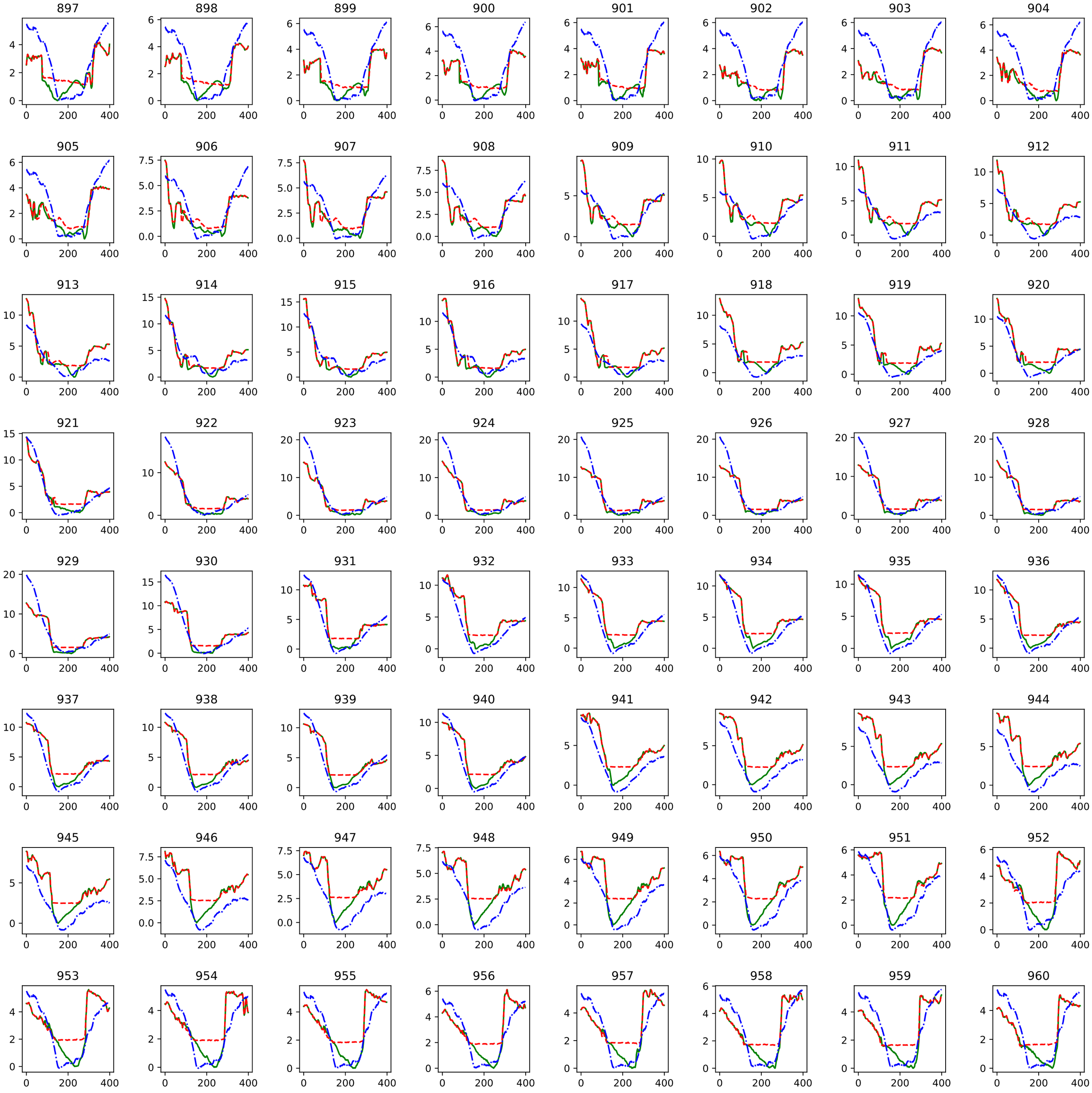

577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640


641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704


705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768


769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832


833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896


897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960

961 962 963 964 965 966 967 968

969 970 971 972 973 974 975 976

977 978 979 980 981 982 983 984

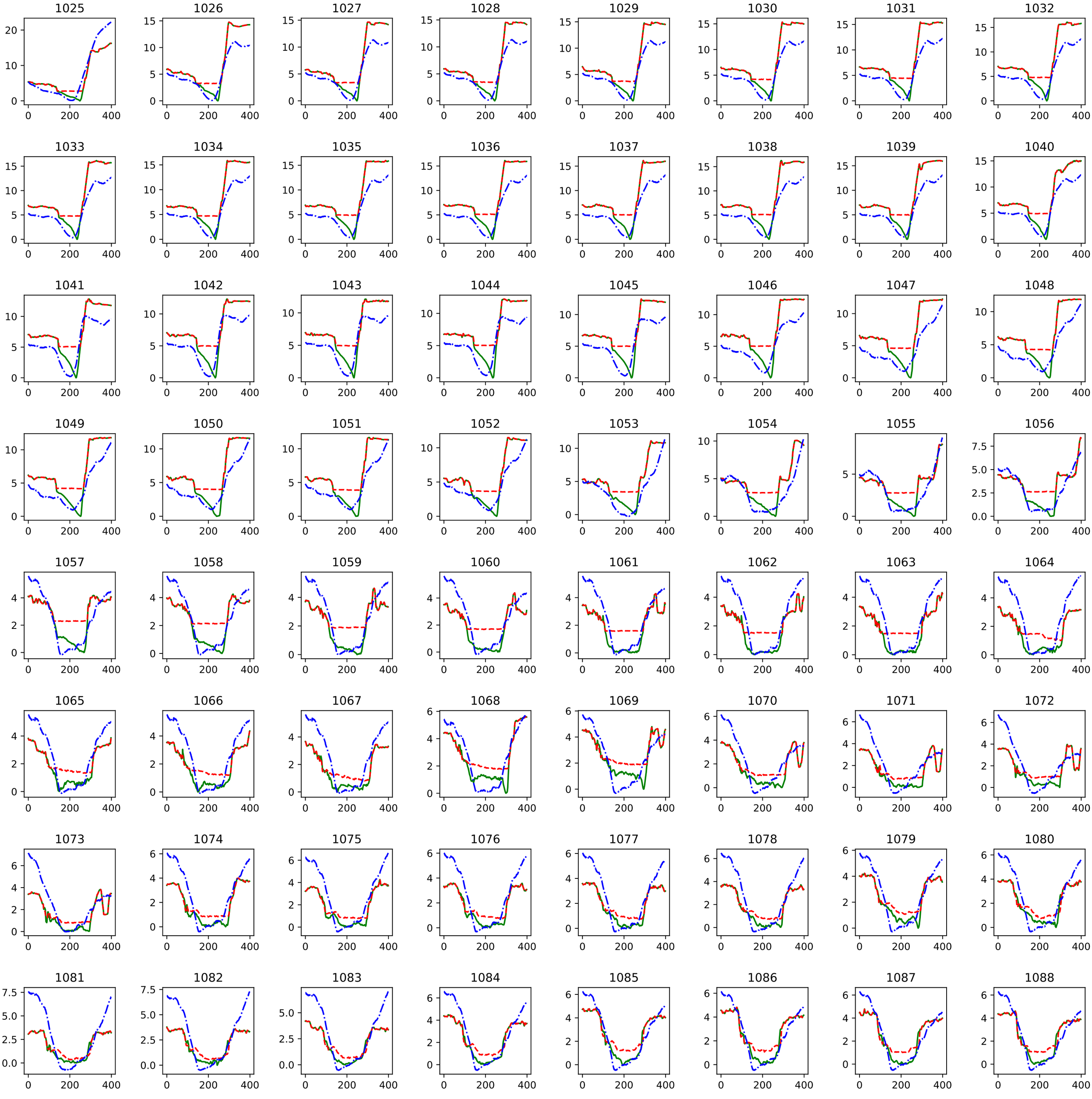
985 986 987 988 989 990 991 992

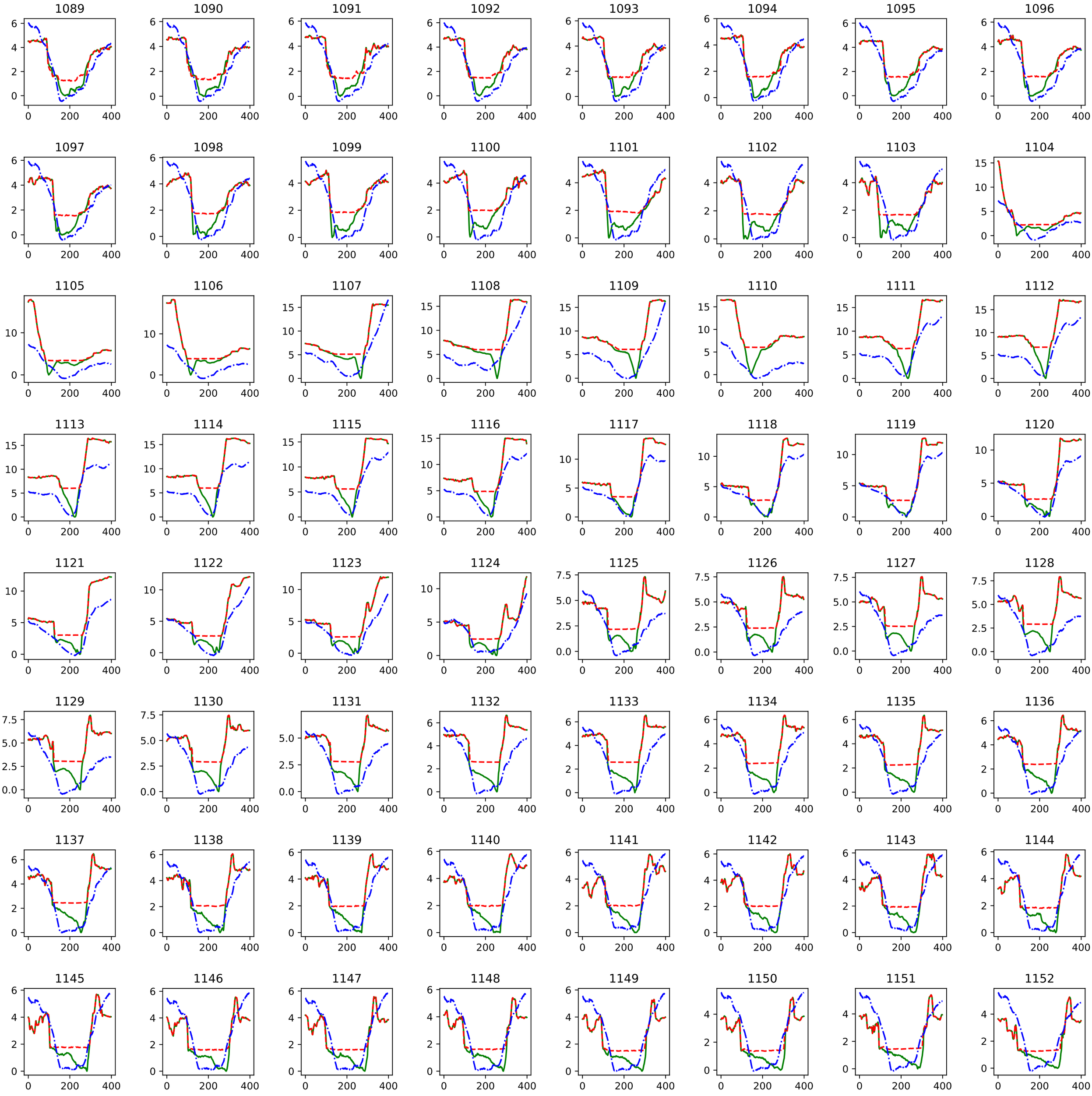
993 994 995 996 997 998 999 1000

1001 1002 1003 1004 1005 1006 1007 1008

1009 1010 1011 1012 1013 1014 1015 1016

1017 1018 1019 1020 1021 1022 1023 1024

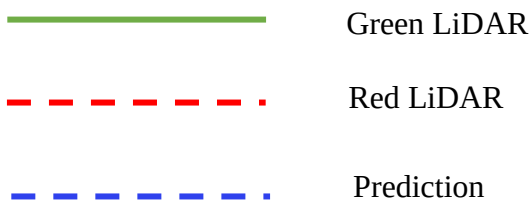

1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088

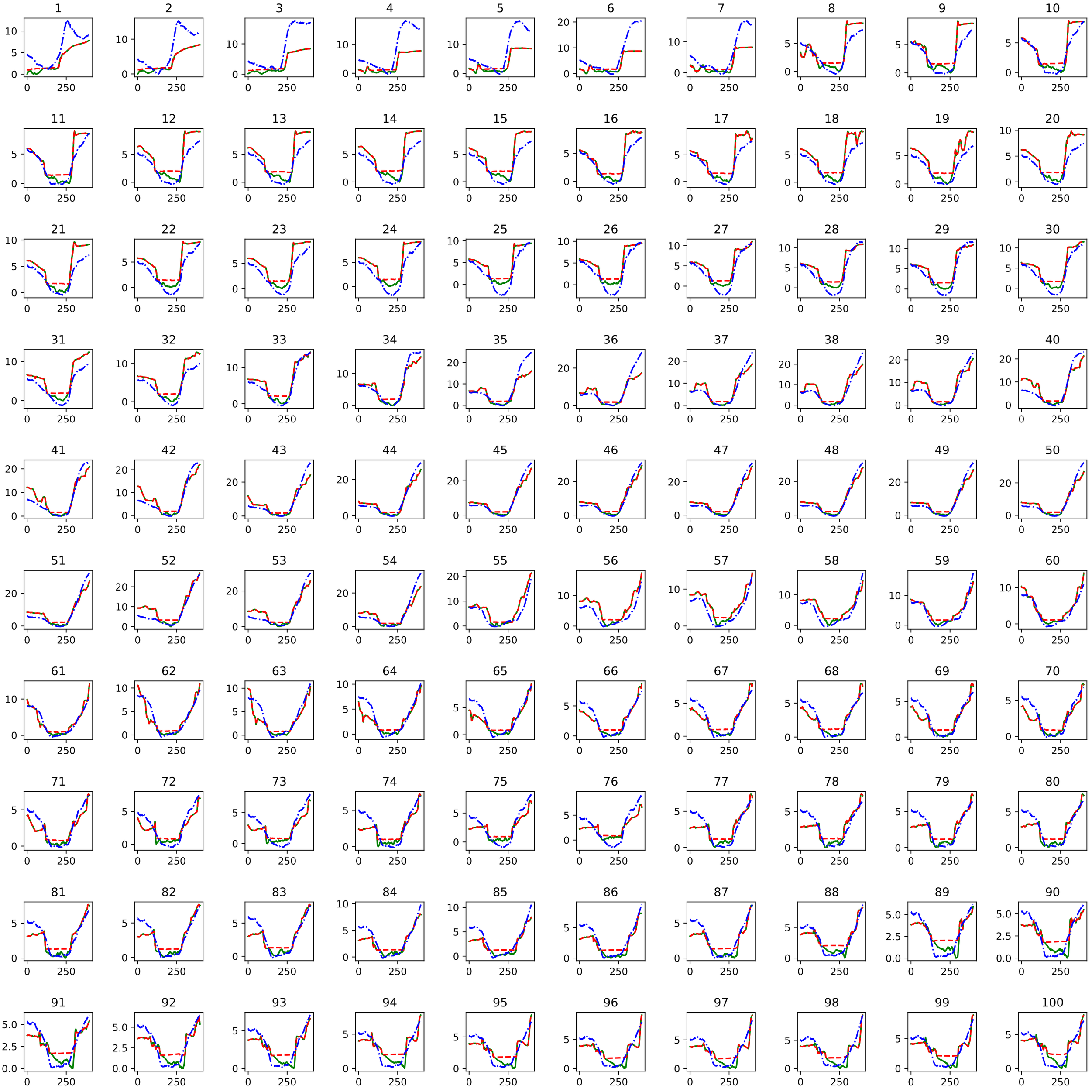

1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152

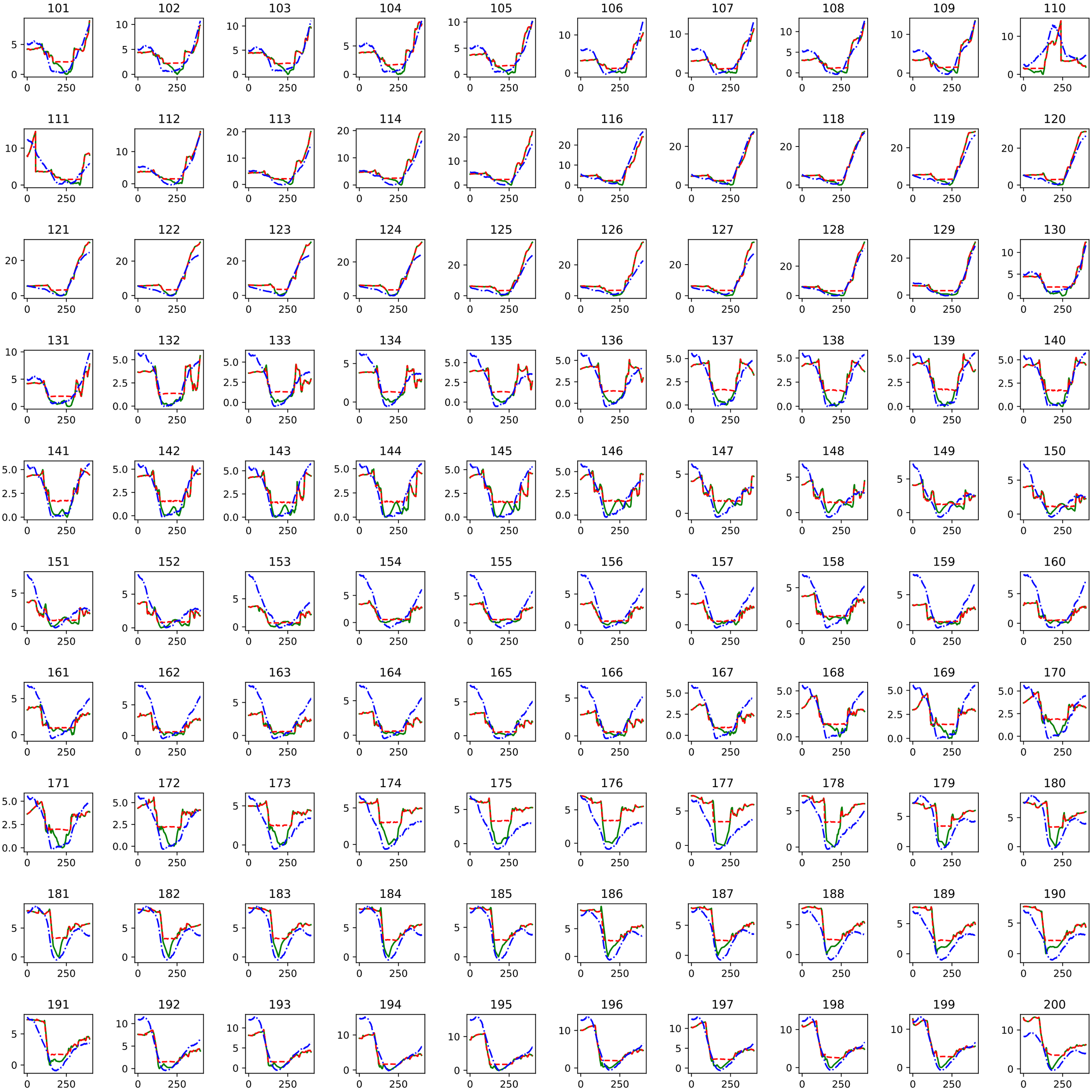
Surna Sigmoid

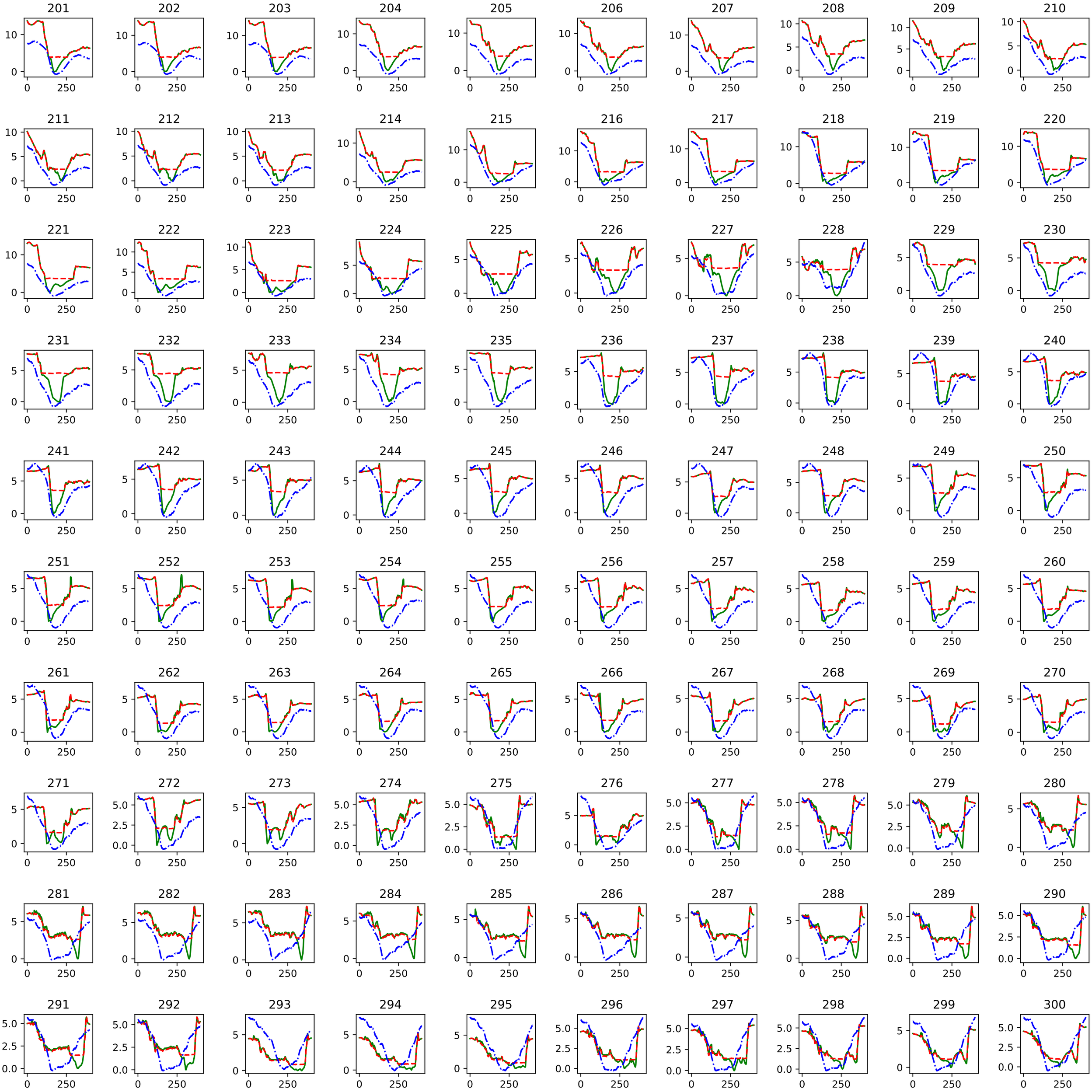
In the next pages, all the cross sections predicted for Surna are shown using Sigmoid activation function.

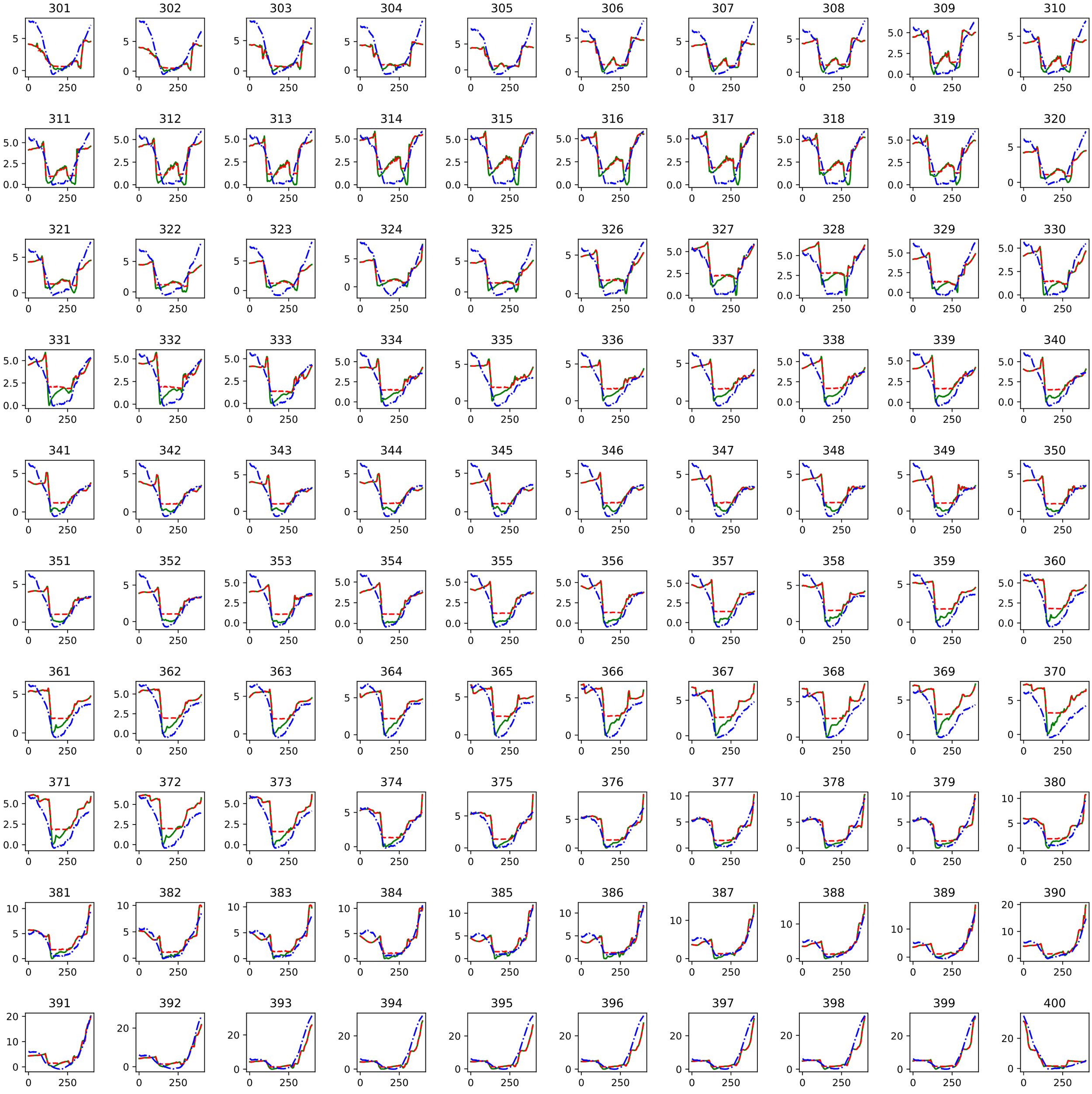
The legend for all the graphs as follow:

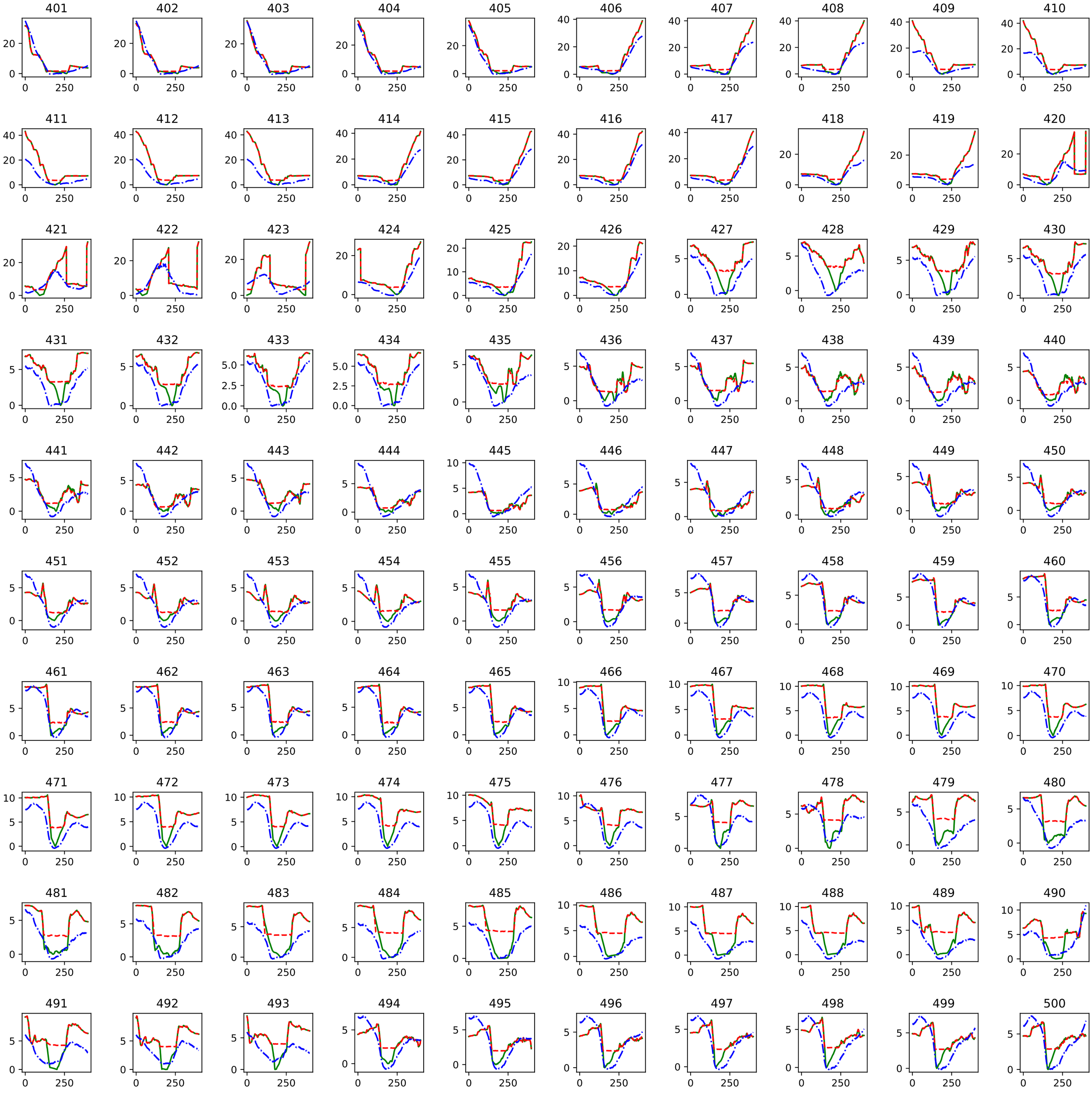



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100


101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200


201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300


301 302 303 304 305 306 307 308 309 310
311 312 313 314 315 316 317 318 319 320
321 322 323 324 325 326 327 328 329 330
331 332 333 334 335 336 337 338 339 340
341 342 343 344 345 346 347 348 349 350
351 352 353 354 355 356 357 358 359 360
361 362 363 364 365 366 367 368 369 370
371 372 373 374 375 376 377 378 379 380
381 382 383 384 385 386 387 388 389 390
391 392 393 394 395 396 397 398 399 400


401 402 403 404 405 406 407 408 409 410
411 412 413 414 415 416 417 418 419 420
421 422 423 424 425 426 427 428 429 430
431 432 433 434 435 436 437 438 439 440
441 442 443 444 445 446 447 448 449 450
451 452 453 454 455 456 457 458 459 460
461 462 463 464 465 466 467 468 469 470
471 472 473 474 475 476 477 478 479 480
481 482 483 484 485 486 487 488 489 490
491 492 493 494 495 496 497 498 499 500

501 502 503 504 505 506 507 508 509 510

511 512 513 514 515 516 517 518 519 520

521 522 523 524 525 526 527 528 529 530

531 532 533 534 535 536 537 538 539 540

541 542 543 544 545 546 547 548 549 550

551 552 553 554 555 556 557 558 559 560

561 562 563 564 565 566 567 568 569 570

571 572 573 574 575 576 577 578 579 580

581 582 583 584 585 586 587 588 589 590

591 592 593 594 595 596 597 598 599 600

601 602 603 604 605 606 607 608 609 610

611 612 613 614 615 616 617 618 619 620

621 622 623 624 625 626 627 628 629 630

631 632 633 634 635 636 637 638 639 640

641 642 643 644 645 646 647 648 649 650

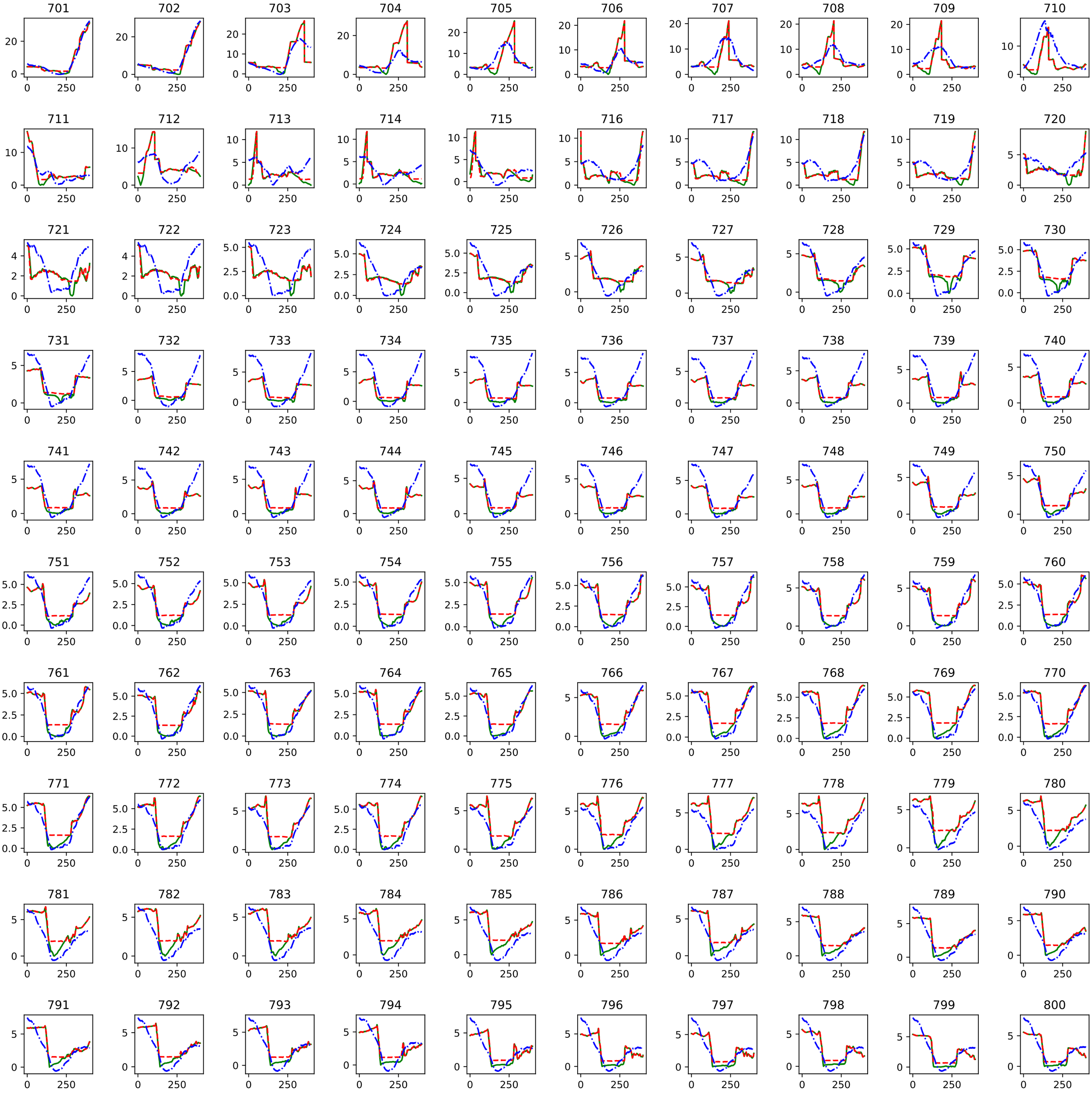
651 652 653 654 655 656 657 658 659 660

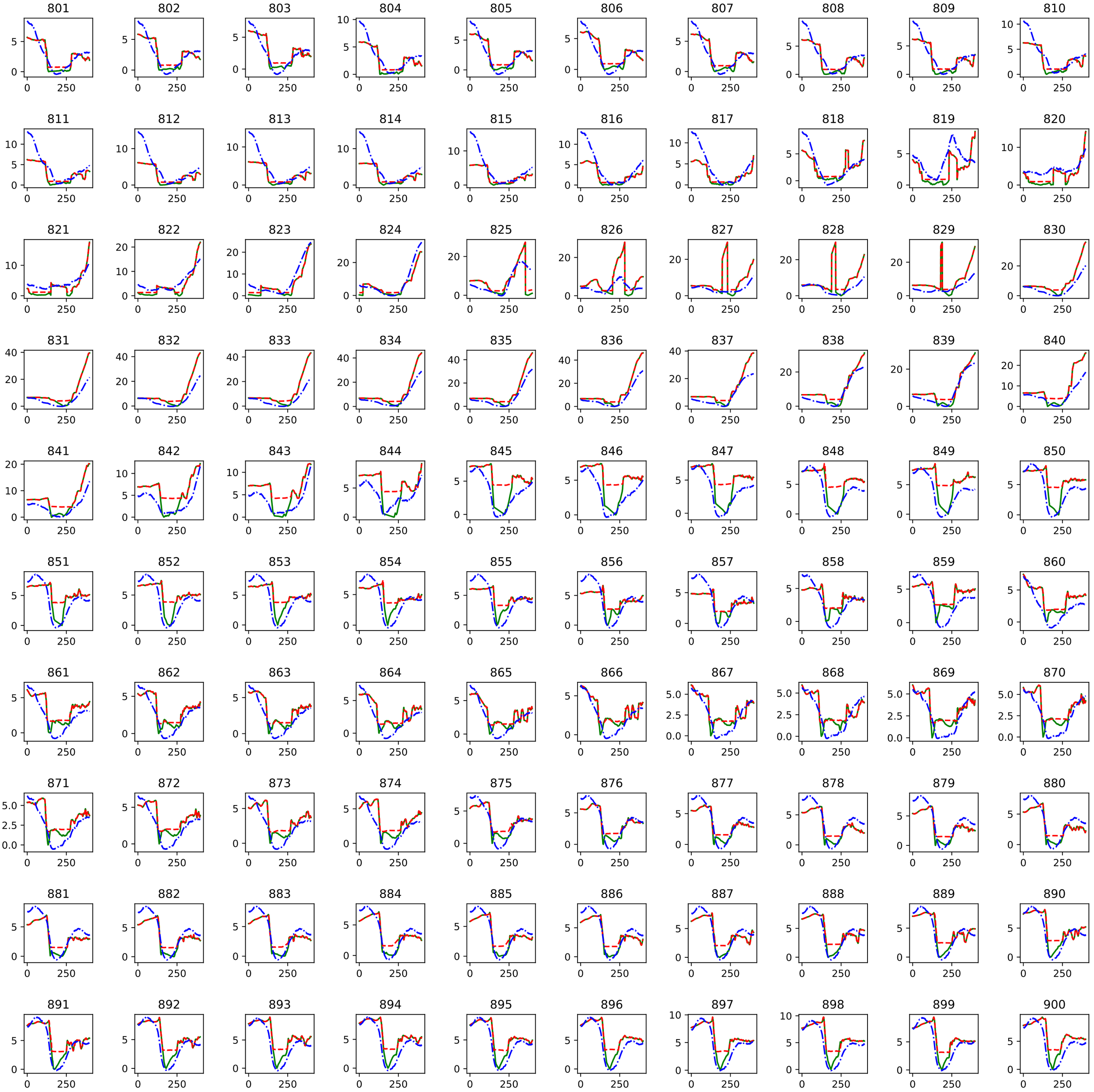
661 662 663 664 665 666 667 668 669 670

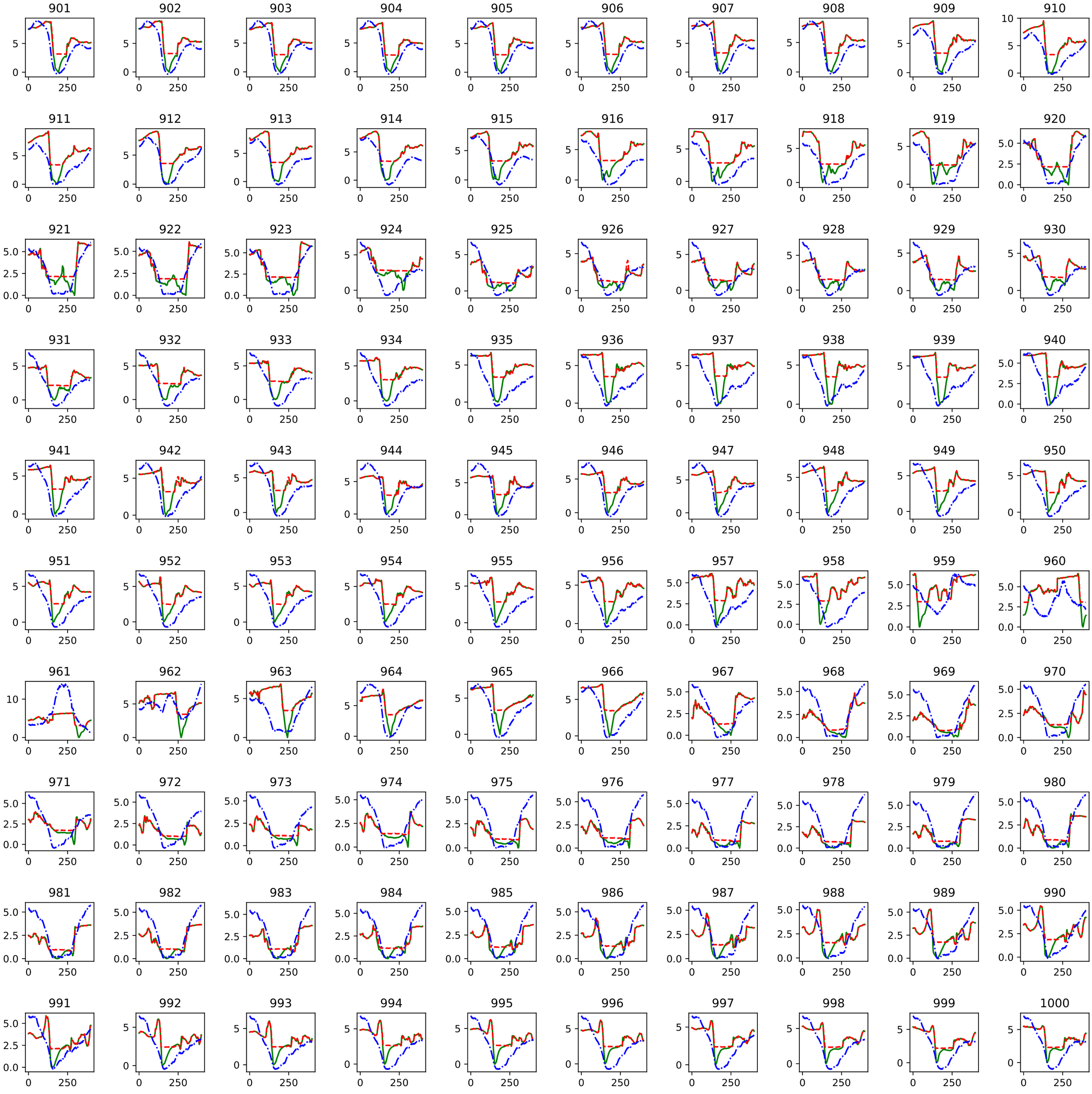
671 672 673 674 675 676 677 678 679 680

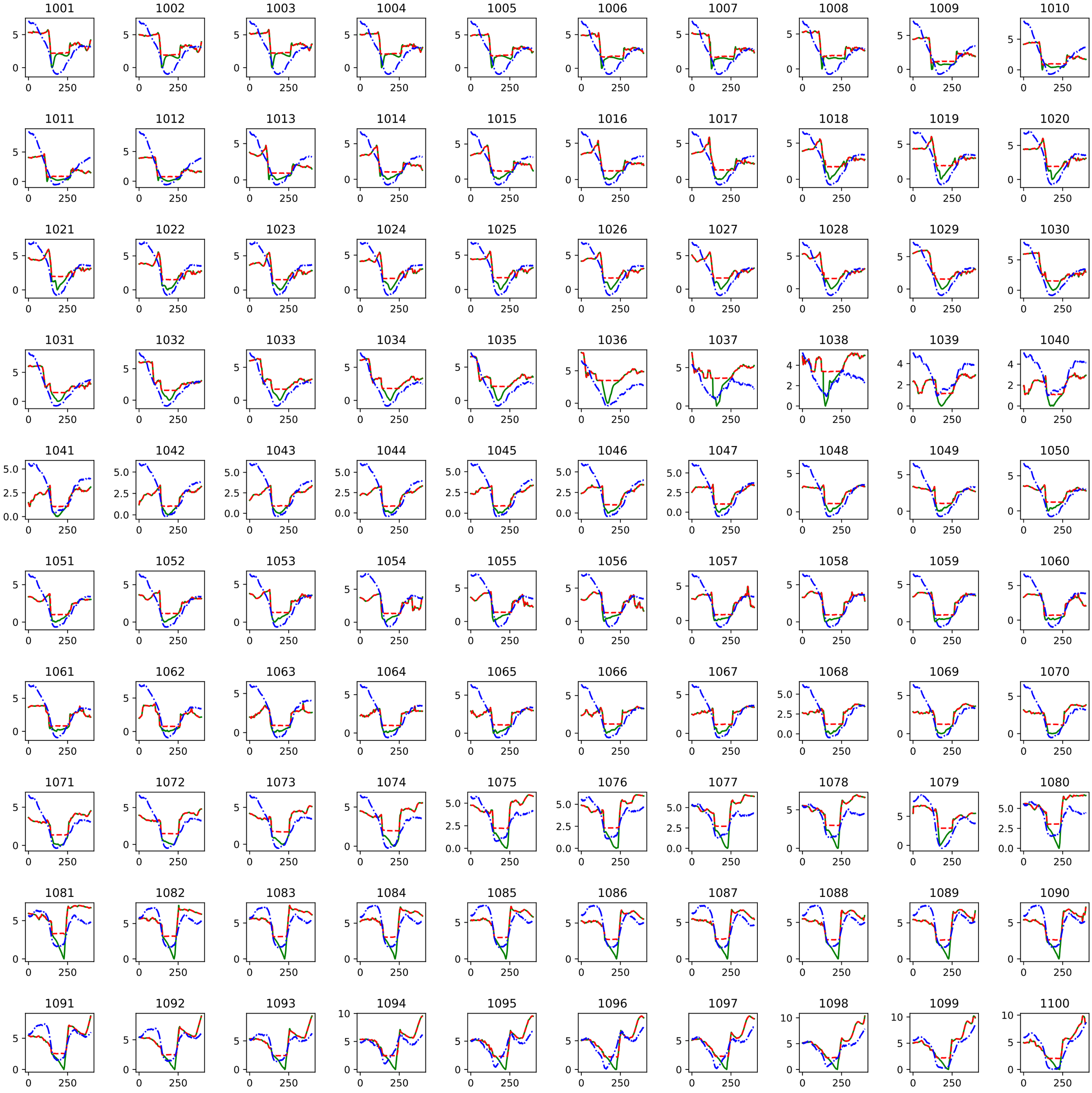
681 682 683 684 685 686 687 688 689 690

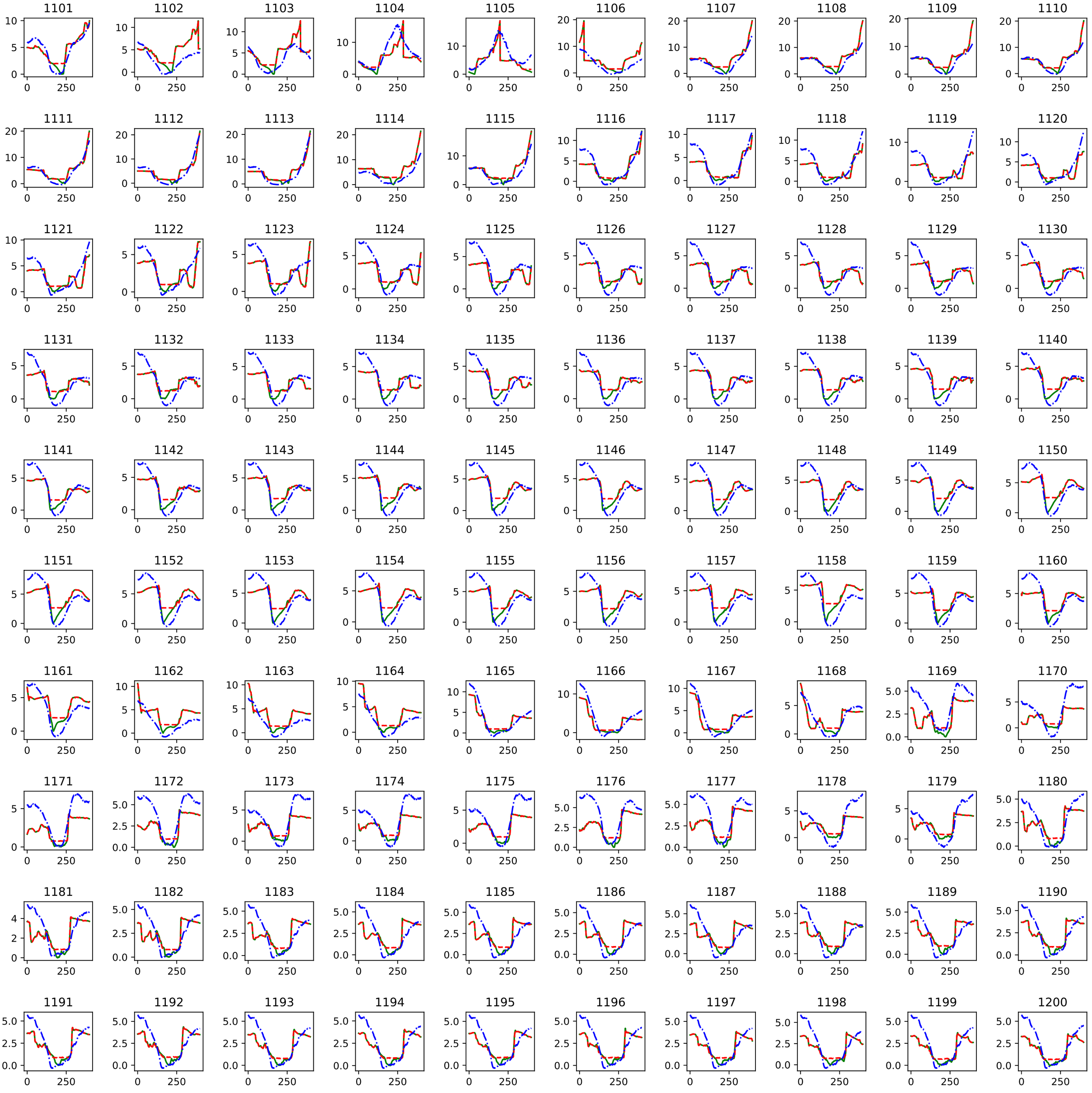
691 692 693 694 695 696 697 698 699 700

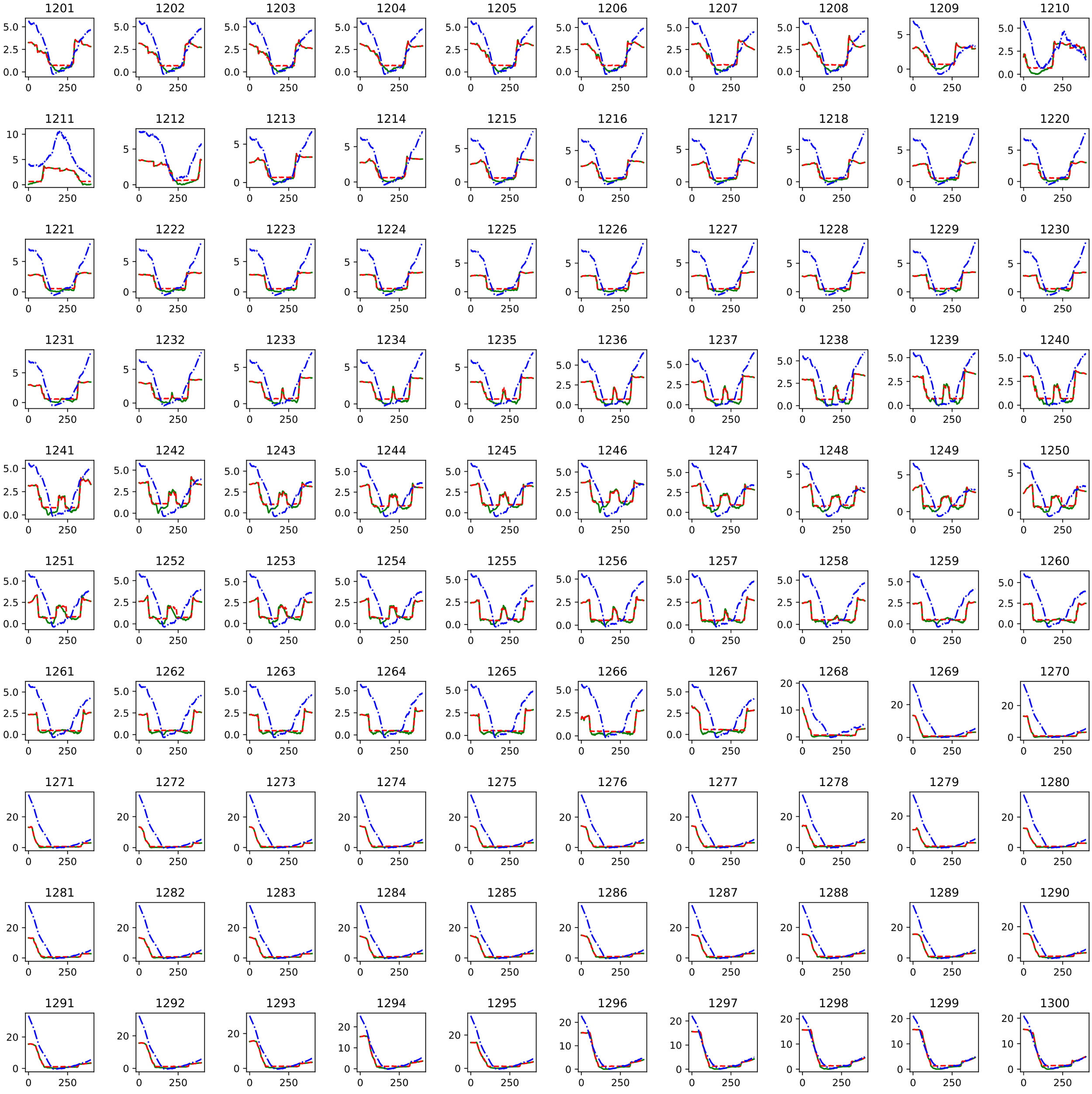

701 702 703 704 705 706 707 708 709 710
711 712 713 714 715 716 717 718 719 720
721 722 723 724 725 726 727 728 729 730
731 732 733 734 735 736 737 738 739 740
741 742 743 744 745 746 747 748 749 750
751 752 753 754 755 756 757 758 759 760
761 762 763 764 765 766 767 768 769 770
771 772 773 774 775 776 777 778 779 780
781 782 783 784 785 786 787 788 789 790
791 792 793 794 795 796 797 798 799 800

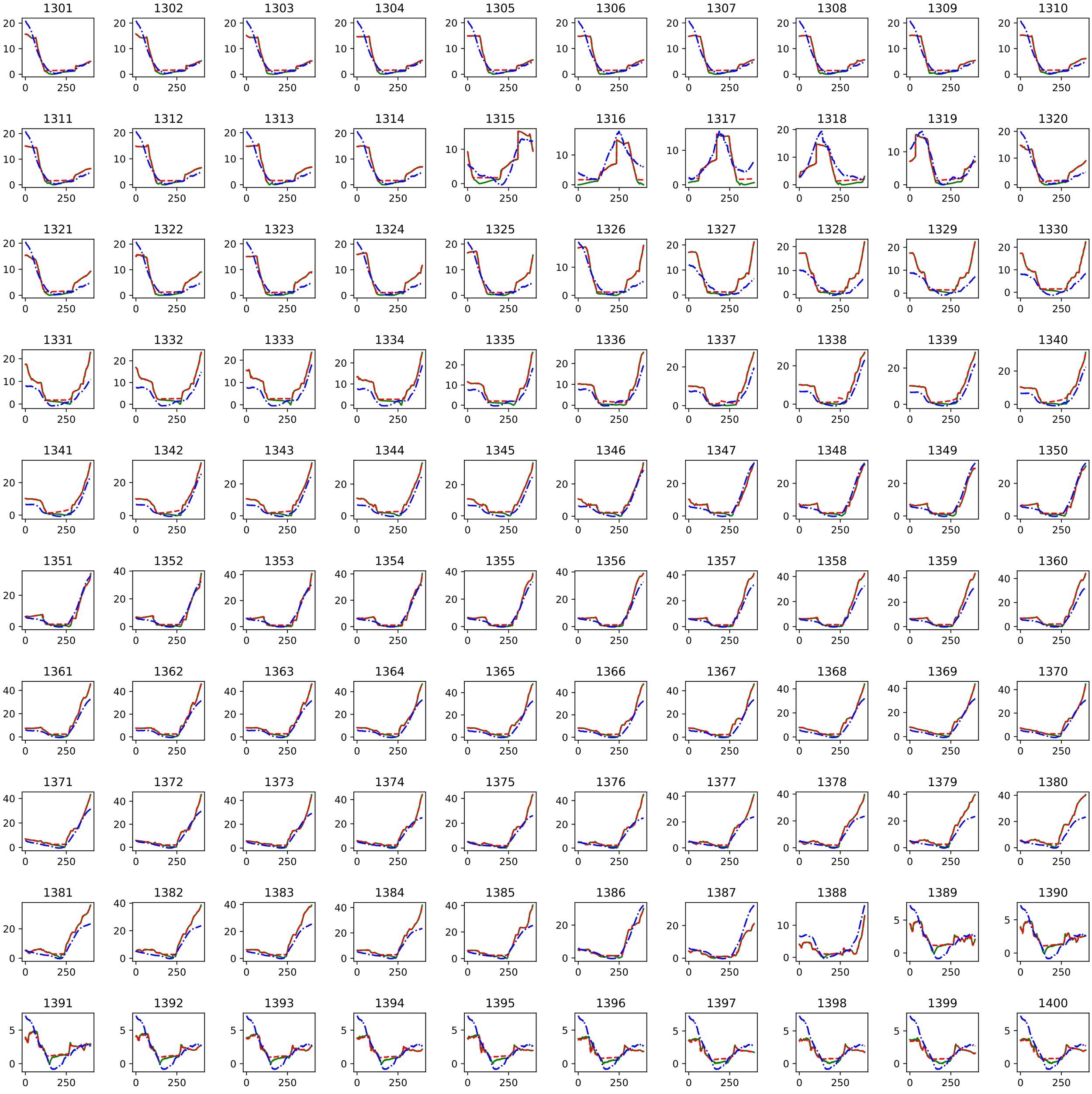

801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900

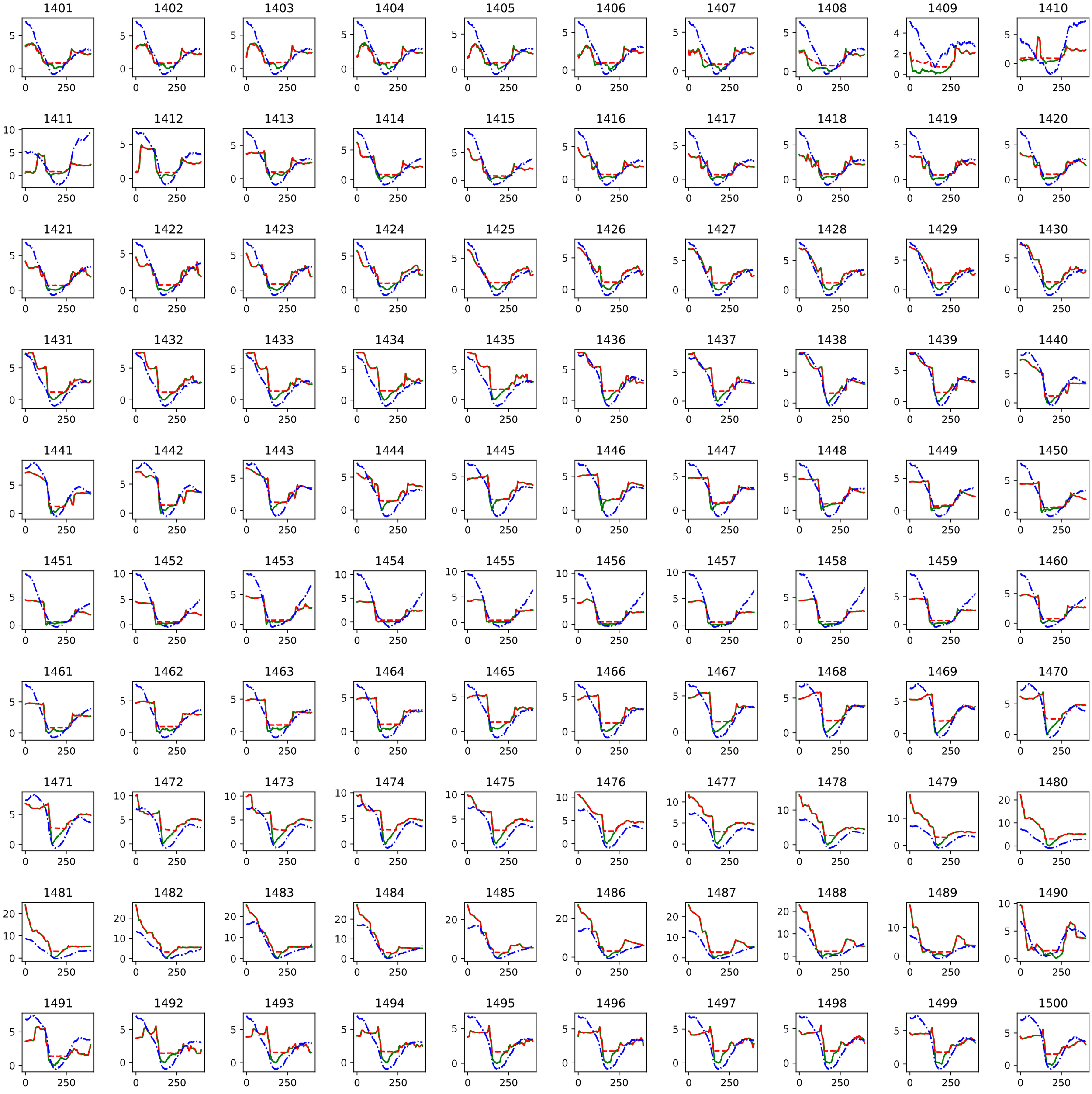

901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

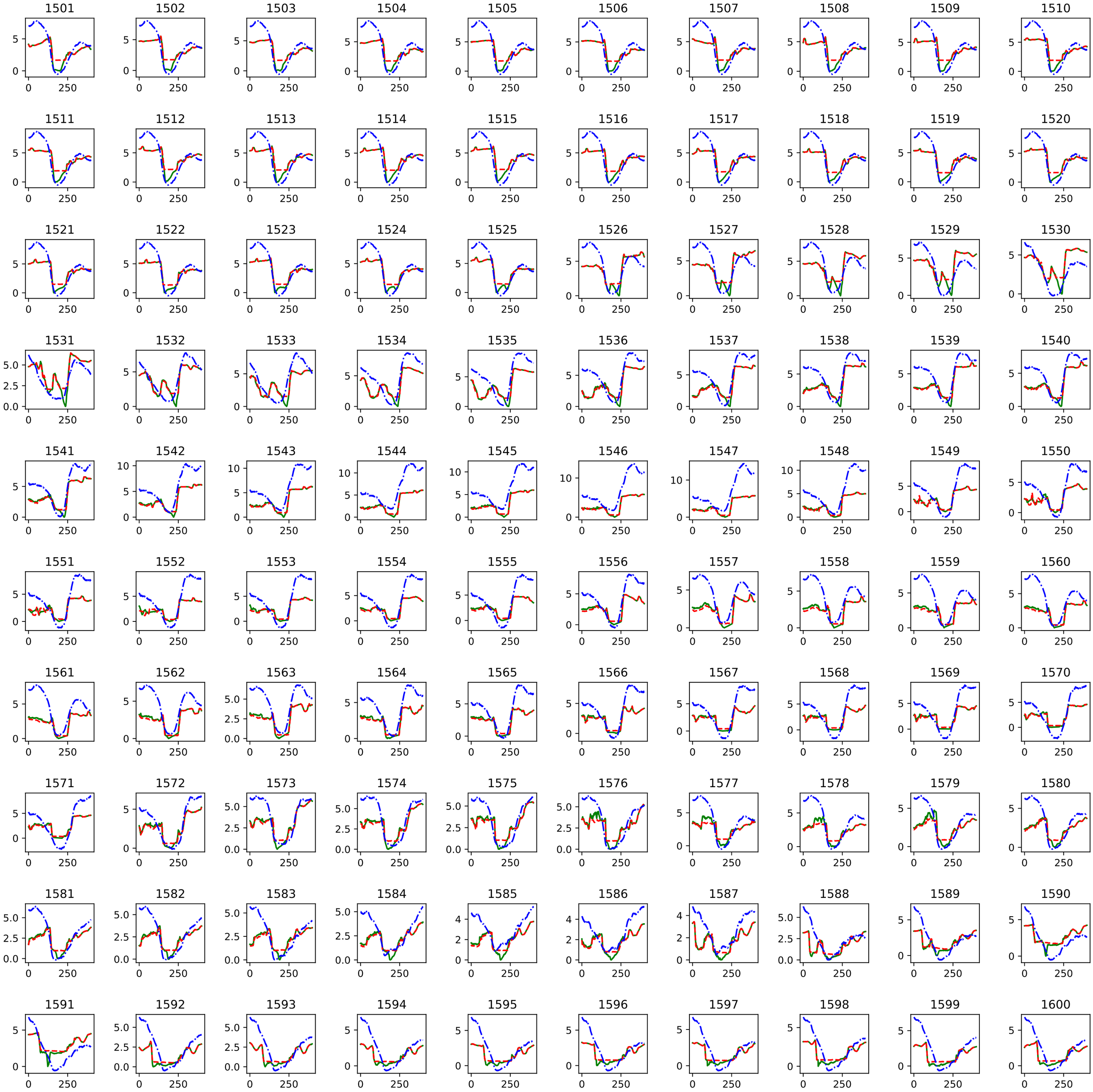

1001 1002 1003 1004 1005 1006 1007 1008 1009 1010
1011 1012 1013 1014 1015 1016 1017 1018 1019 1020
1021 1022 1023 1024 1025 1026 1027 1028 1029 1030
1031 1032 1033 1034 1035 1036 1037 1038 1039 1040
1041 1042 1043 1044 1045 1046 1047 1048 1049 1050
1051 1052 1053 1054 1055 1056 1057 1058 1059 1060
1061 1062 1063 1064 1065 1066 1067 1068 1069 1070
1071 1072 1073 1074 1075 1076 1077 1078 1079 1080
1081 1082 1083 1084 1085 1086 1087 1088 1089 1090
1091 1092 1093 1094 1095 1096 1097 1098 1099 1100

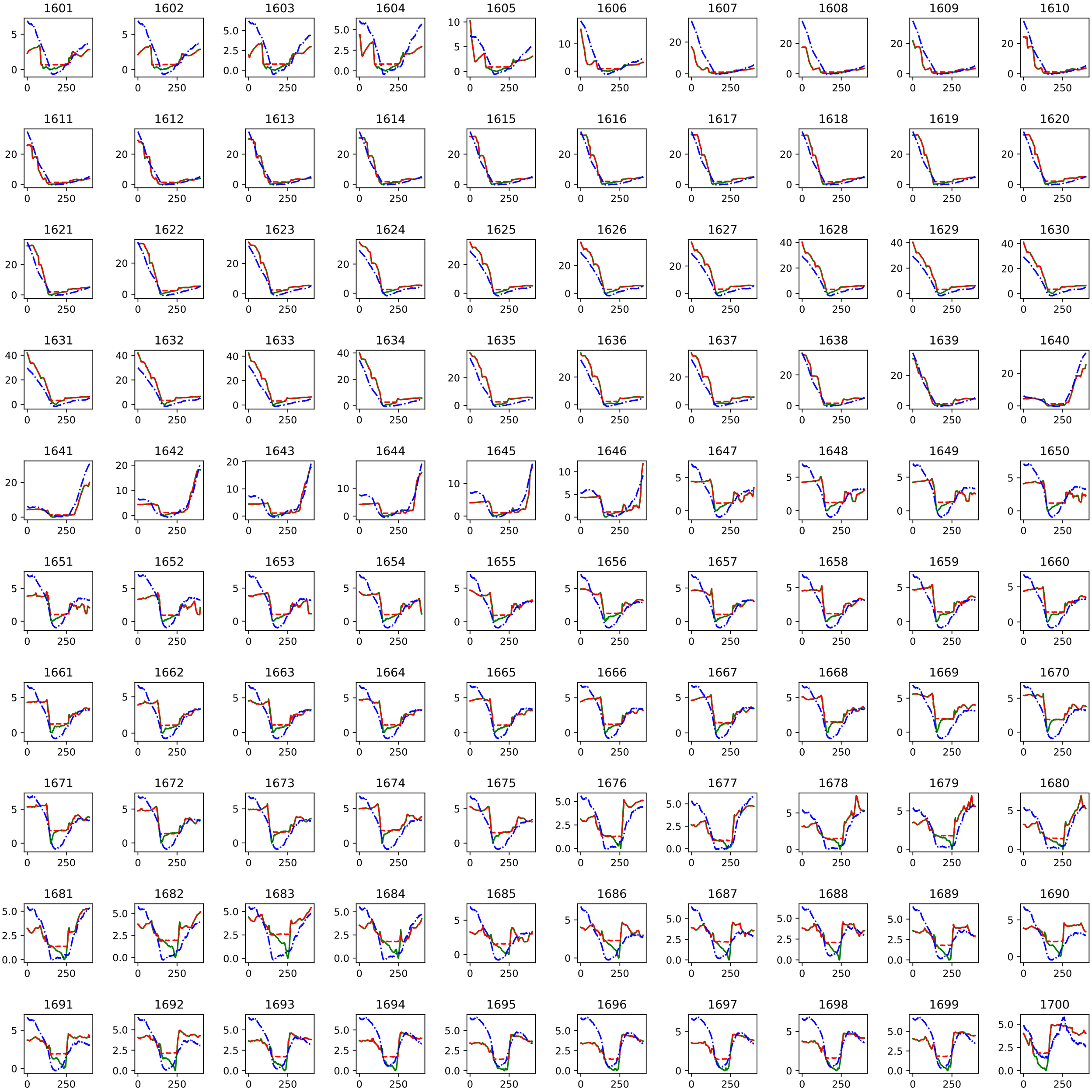

1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200

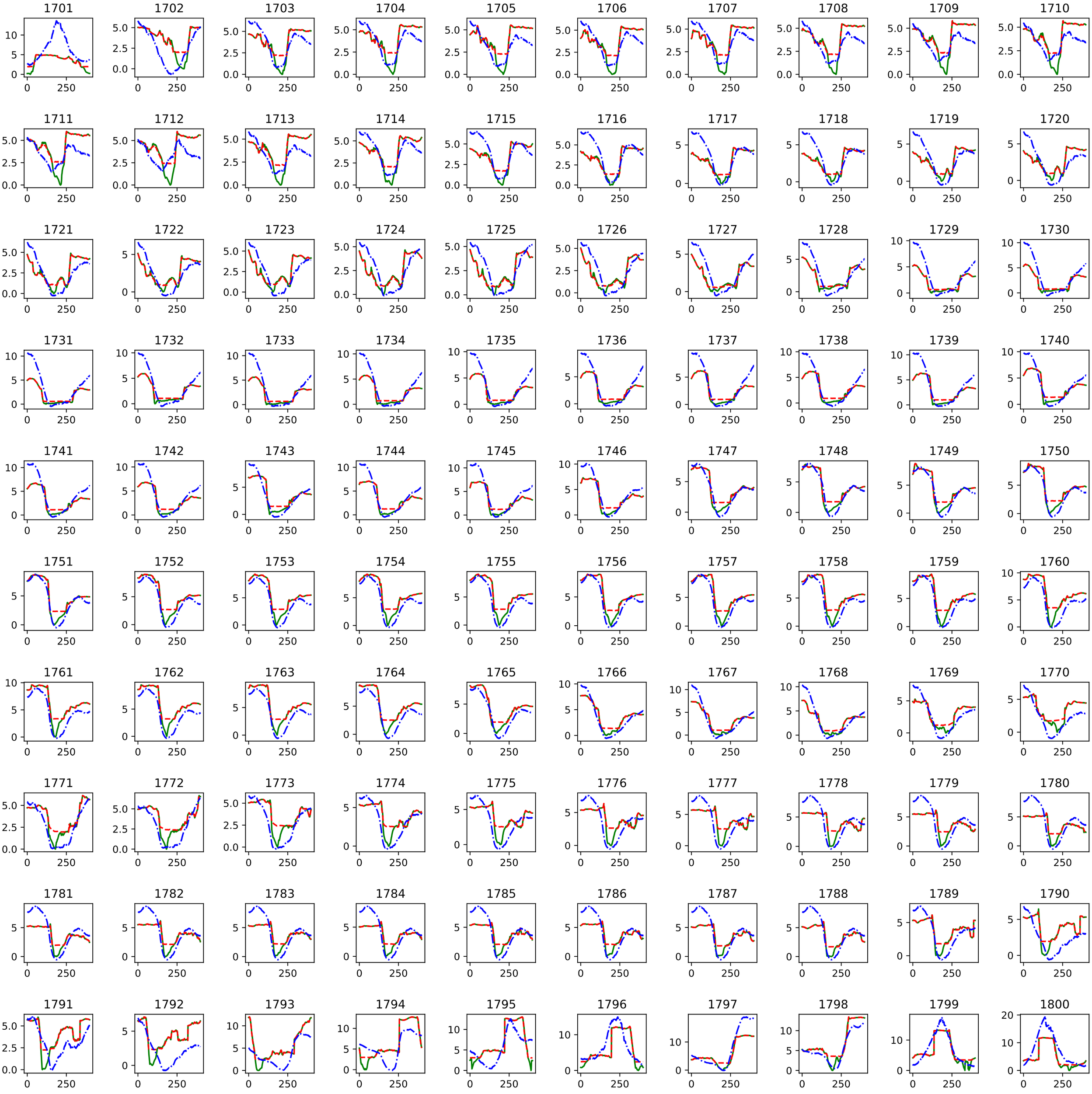

1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300


1301 1302 1303 1304 1305 1306 1307 1308 1309 1310
1311 1312 1313 1314 1315 1316 1317 1318 1319 1320
1321 1322 1323 1324 1325 1326 1327 1328 1329 1330
1331 1332 1333 1334 1335 1336 1337 1338 1339 1340
1341 1342 1343 1344 1345 1346 1347 1348 1349 1350
1351 1352 1353 1354 1355 1356 1357 1358 1359 1360
1361 1362 1363 1364 1365 1366 1367 1368 1369 1370
1371 1372 1373 1374 1375 1376 1377 1378 1379 1380
1381 1382 1383 1384 1385 1386 1387 1388 1389 1390
1391 1392 1393 1394 1395 1396 1397 1398 1399 1400


1401 1402 1403 1404 1405 1406 1407 1408 1409 1410
1411 1412 1413 1414 1415 1416 1417 1418 1419 1420
1421 1422 1423 1424 1425 1426 1427 1428 1429 1430
1431 1432 1433 1434 1435 1436 1437 1438 1439 1440
1441 1442 1443 1444 1445 1446 1447 1448 1449 1450
1451 1452 1453 1454 1455 1456 1457 1458 1459 1460
1461 1462 1463 1464 1465 1466 1467 1468 1469 1470
1471 1472 1473 1474 1475 1476 1477 1478 1479 1480
1481 1482 1483 1484 1485 1486 1487 1488 1489 1490
1491 1492 1493 1494 1495 1496 1497 1498 1499 1500


1501 1502 1503 1504 1505 1506 1507 1508 1509 1510
1511 1512 1513 1514 1515 1516 1517 1518 1519 1520
1521 1522 1523 1524 1525 1526 1527 1528 1529 1530
1531 1532 1533 1534 1535 1536 1537 1538 1539 1540
1541 1542 1543 1544 1545 1546 1547 1548 1549 1550
1551 1552 1553 1554 1555 1556 1557 1558 1559 1560
1561 1562 1563 1564 1565 1566 1567 1568 1569 1570
1571 1572 1573 1574 1575 1576 1577 1578 1579 1580
1581 1582 1583 1584 1585 1586 1587 1588 1589 1590
1591 1592 1593 1594 1595 1596 1597 1598 1599 1600

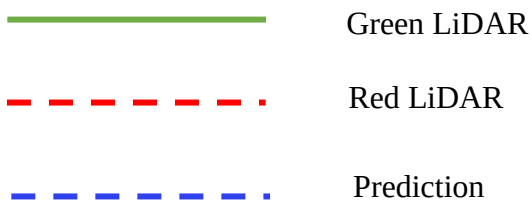

1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700

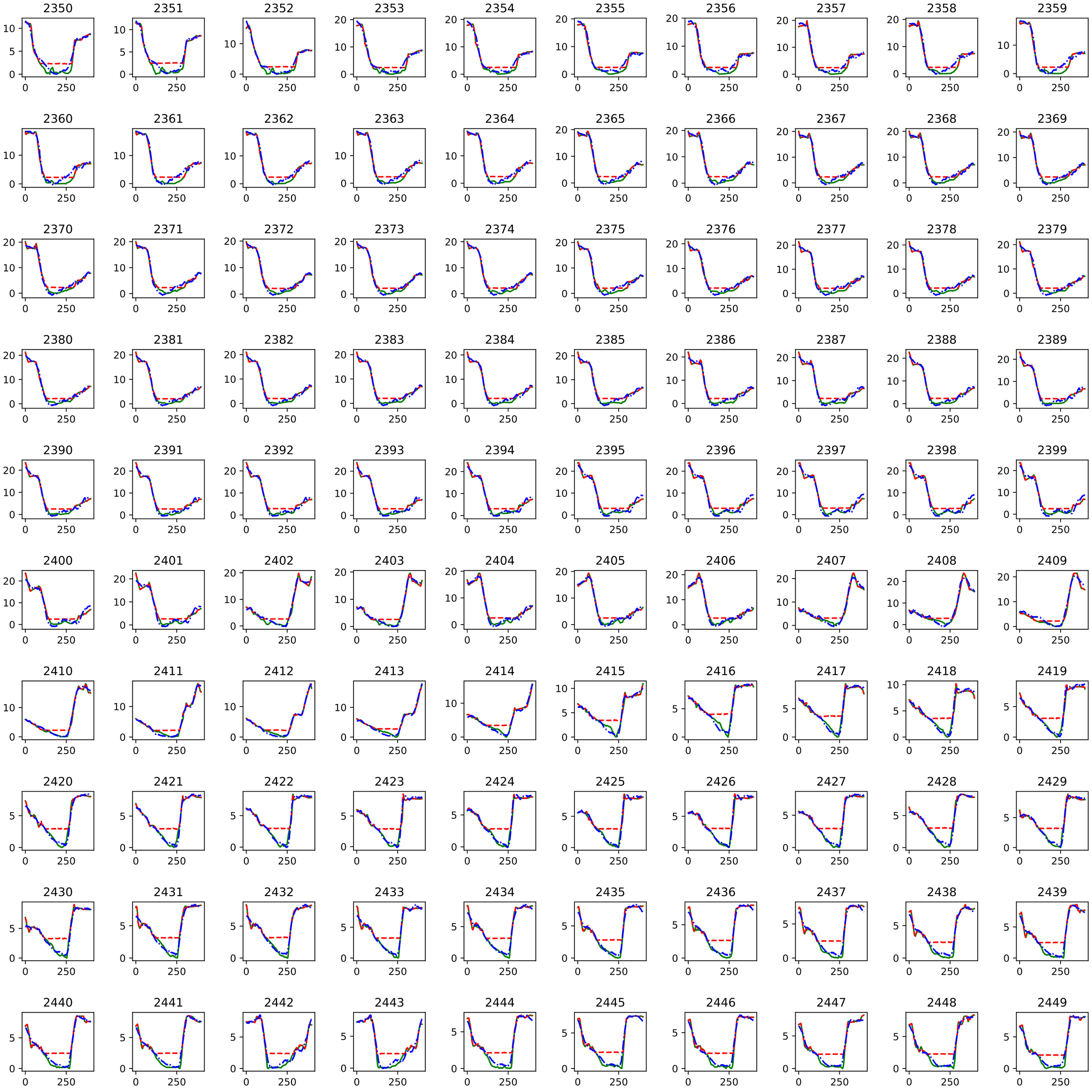

1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800

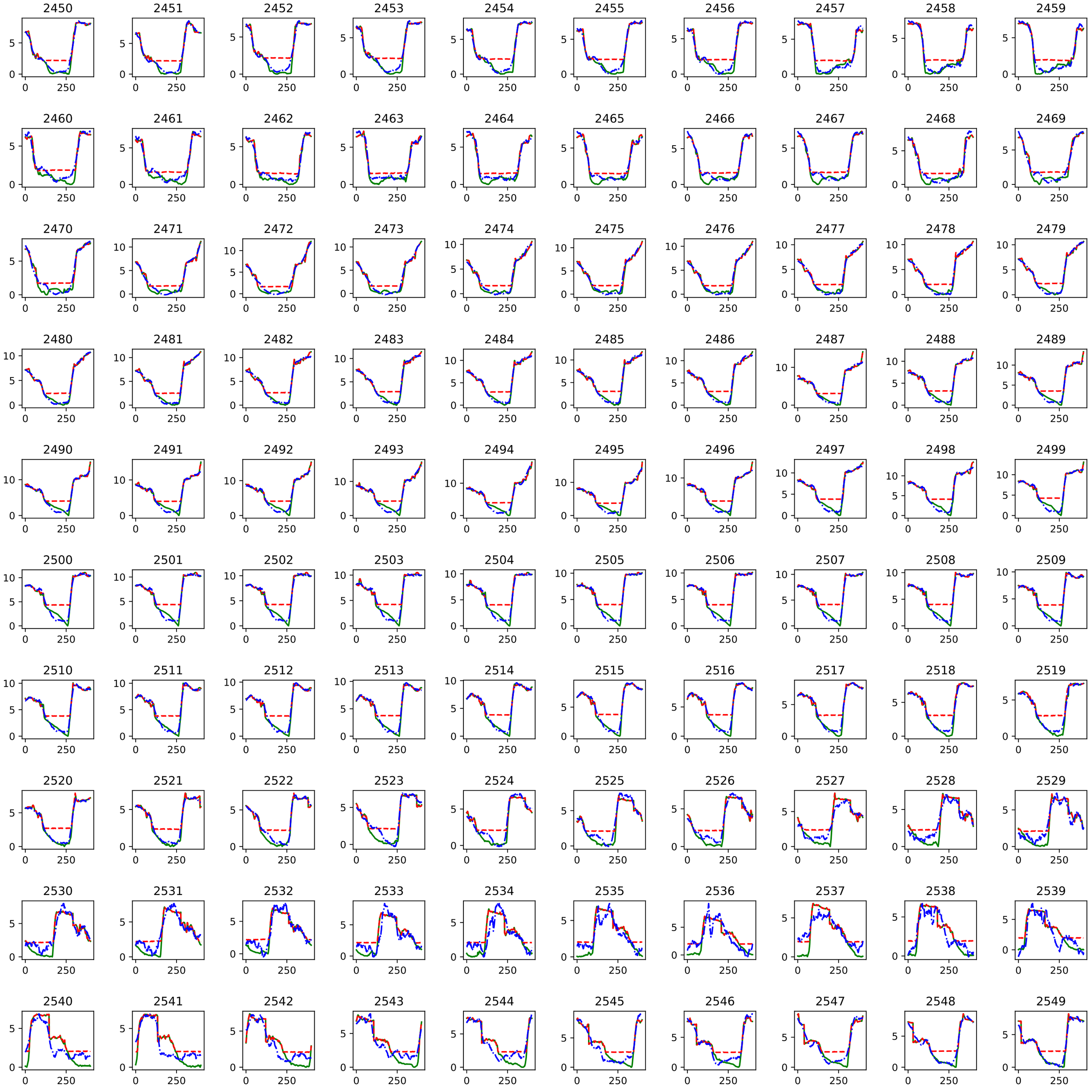
Gaula ReLU

In the next pages, all the cross sections predicted for Gaula outside the range of traning are shown using ReLU activation function.

The legend for all the graphs as follow:



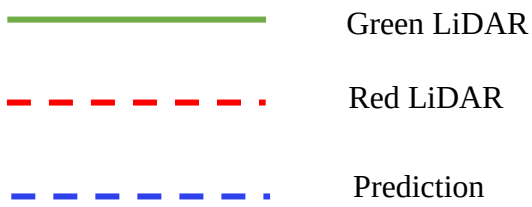

2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449

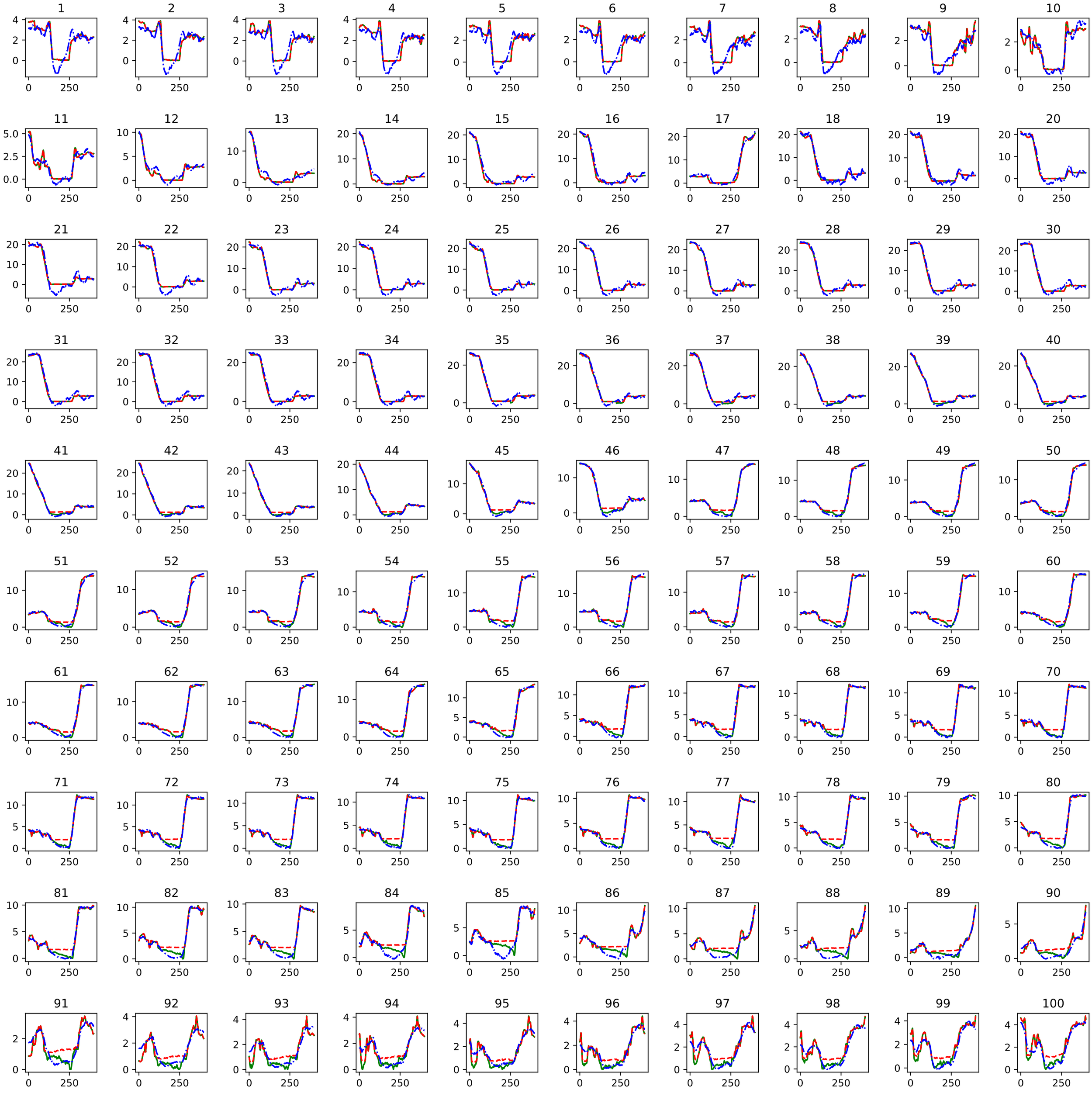

2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549

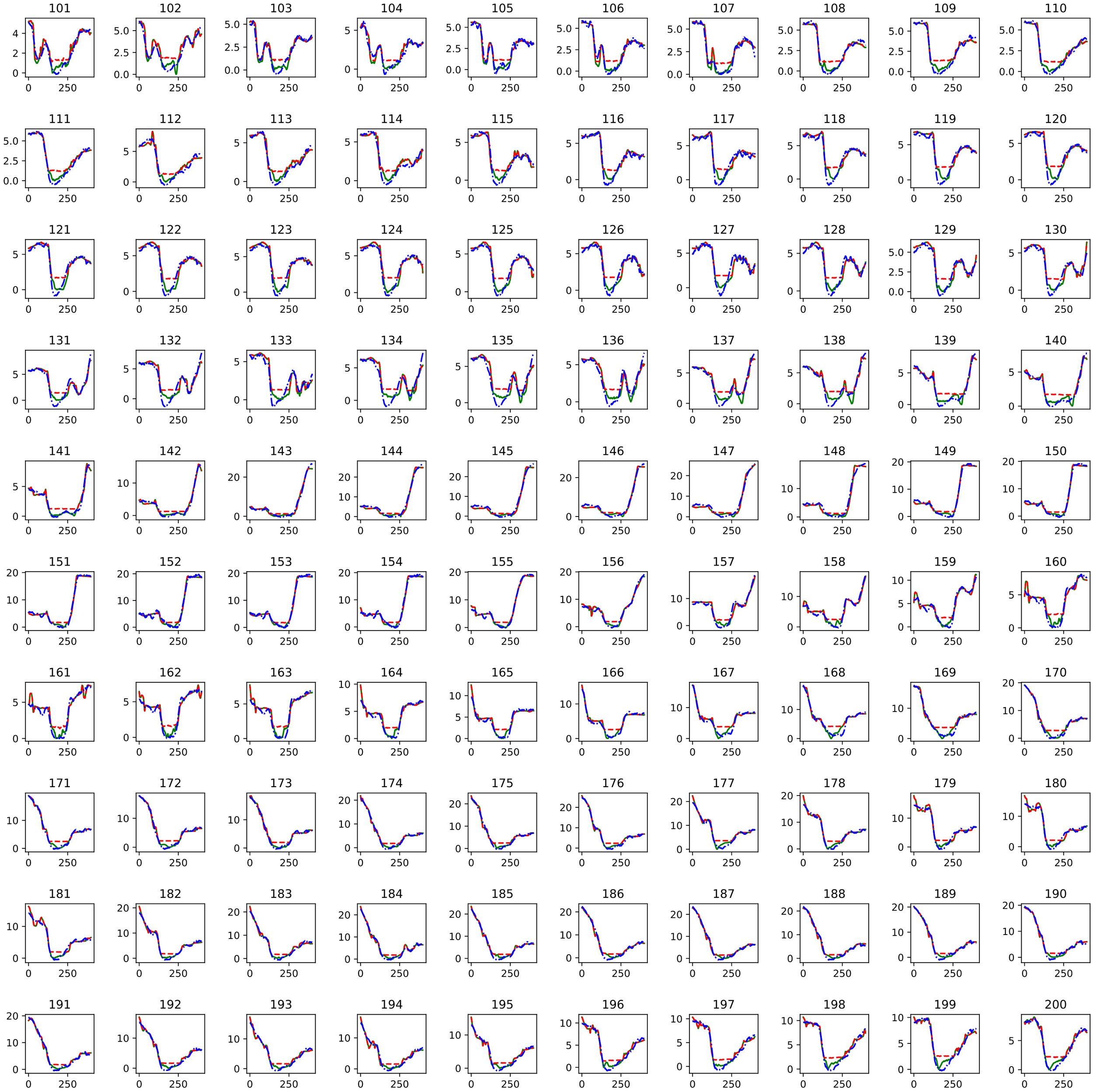
Driva ReLU

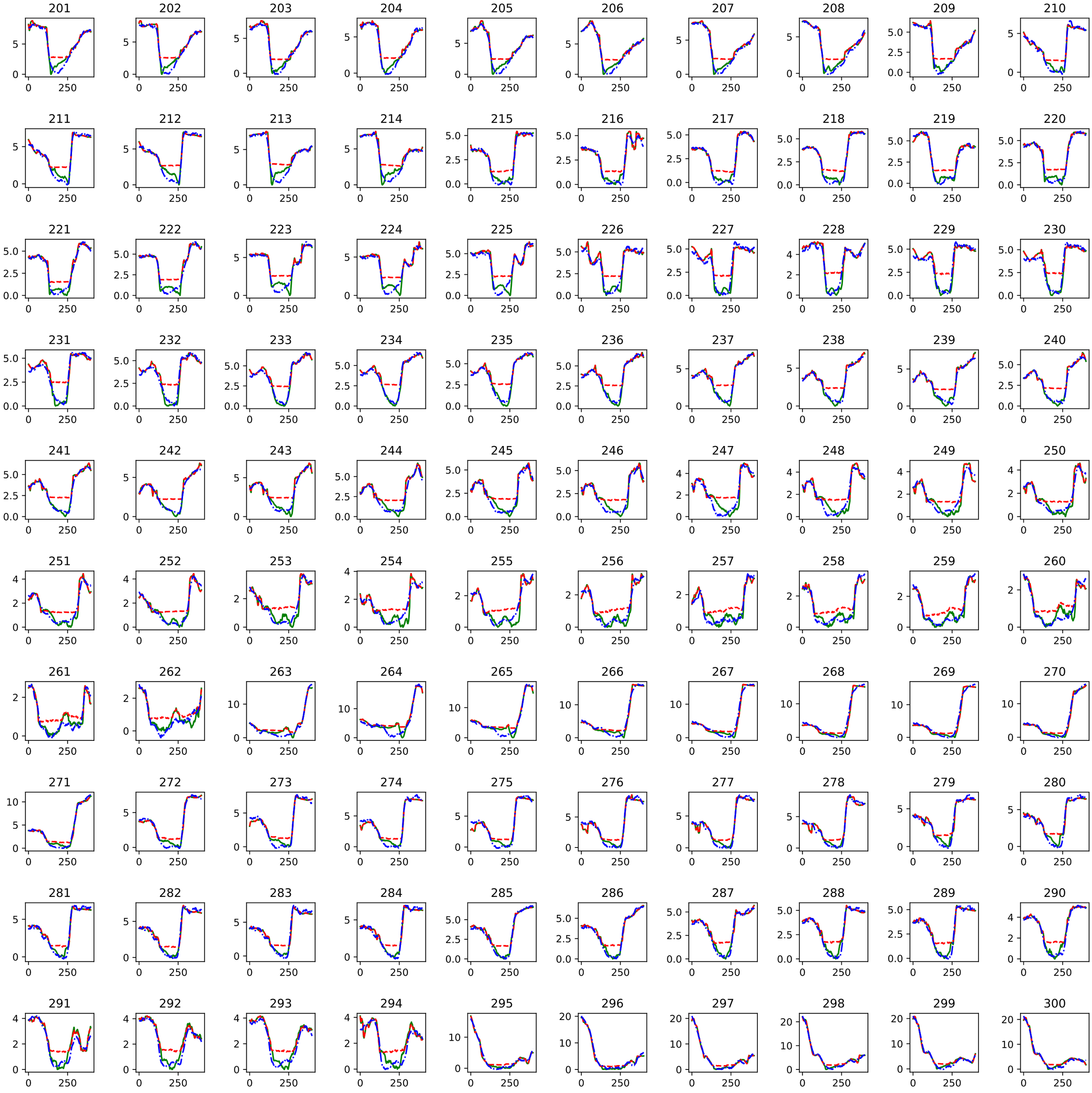
In the next pages, all the cross sections predicted for Driva are shown using ReLU activation function.

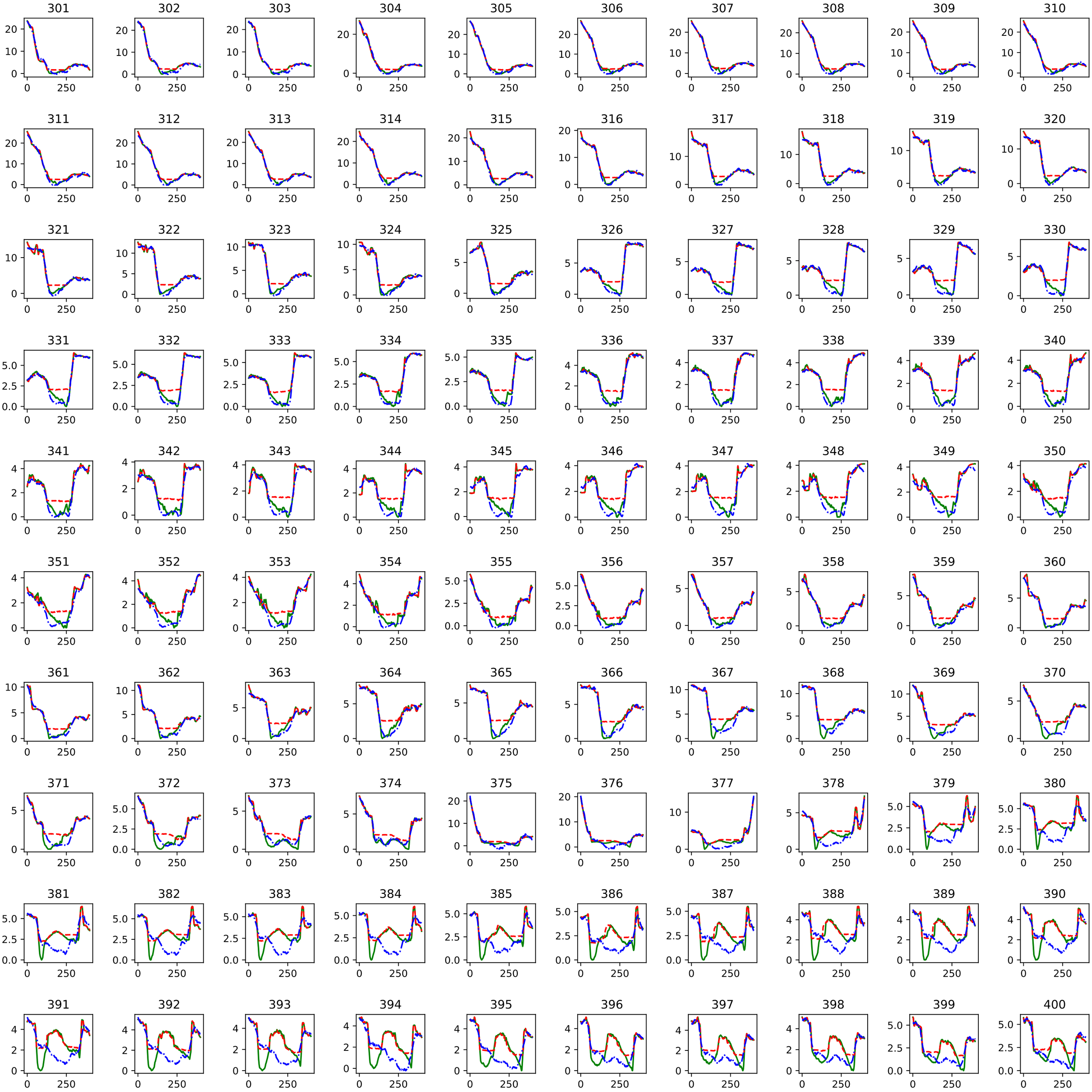
The legend for all the graphs as follow:

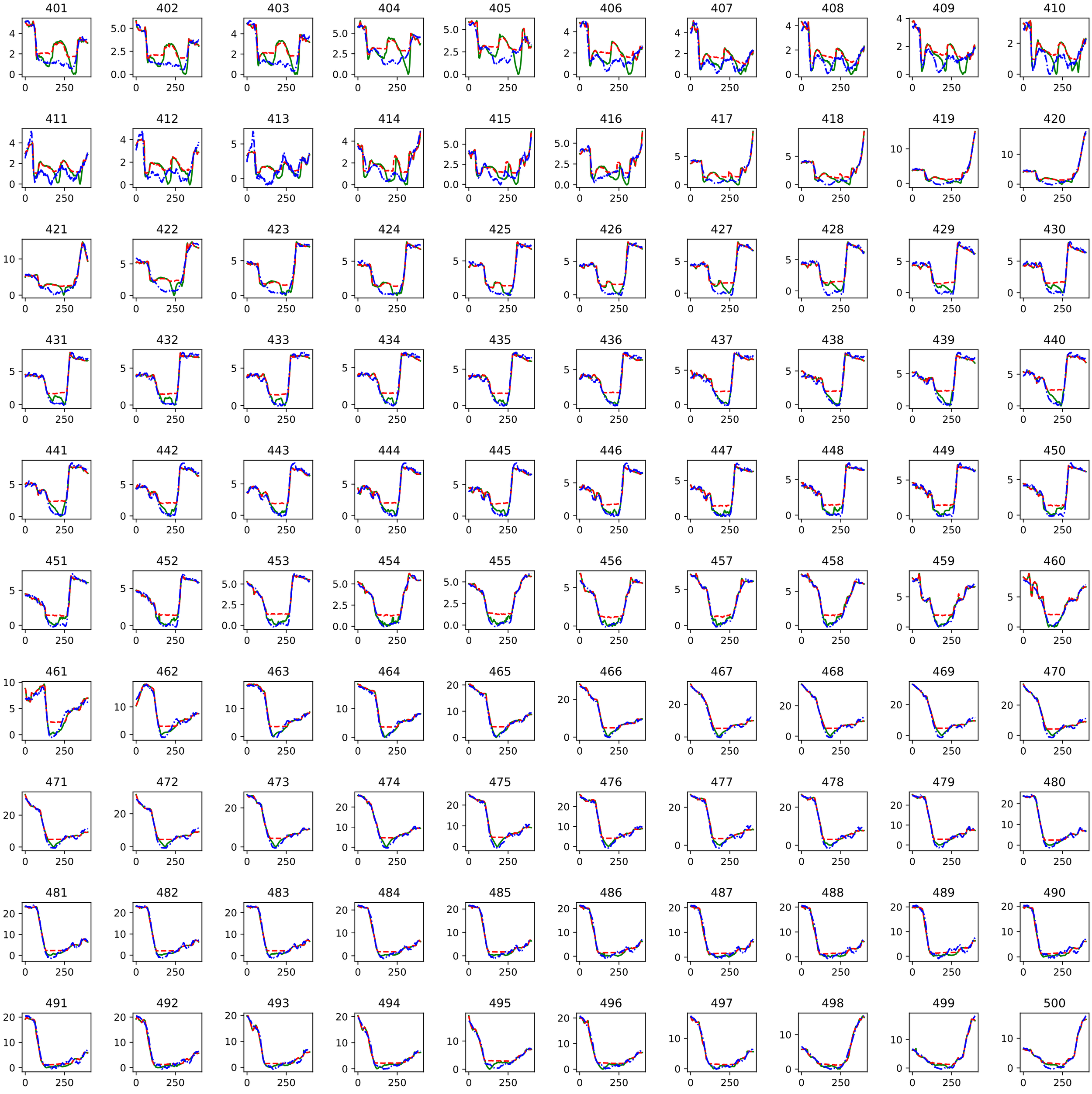



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100


101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200


201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300


301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400


401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500

501 502 503 504 505 506 507 508 509 510
511 512 513 514 515 516 517 518 519 520
521 522 523 524 525 526 527 528 529 530
531 532 533 534 535 536 537 538 539 540
541 542 543 544 545 546 547 548 549 550
551 552 553 554 555 556 557 558 559 560
561 562 563 564 565 566 567 568 569 570
571 572 573 574 575 576 577 578 579 580
581 582 583 584 585 586 587 588 589 590
591 592 593 594 595 596 597 598 599 600

601 602 603 604 605 606 607 608 609 610

611 612 613 614 615 616 617 618 619 620

621 622 623 624 625 626 627 628 629 630

631 632 633 634 635 636 637 638 639 640

641 642 643 644 645 646 647 648 649 650

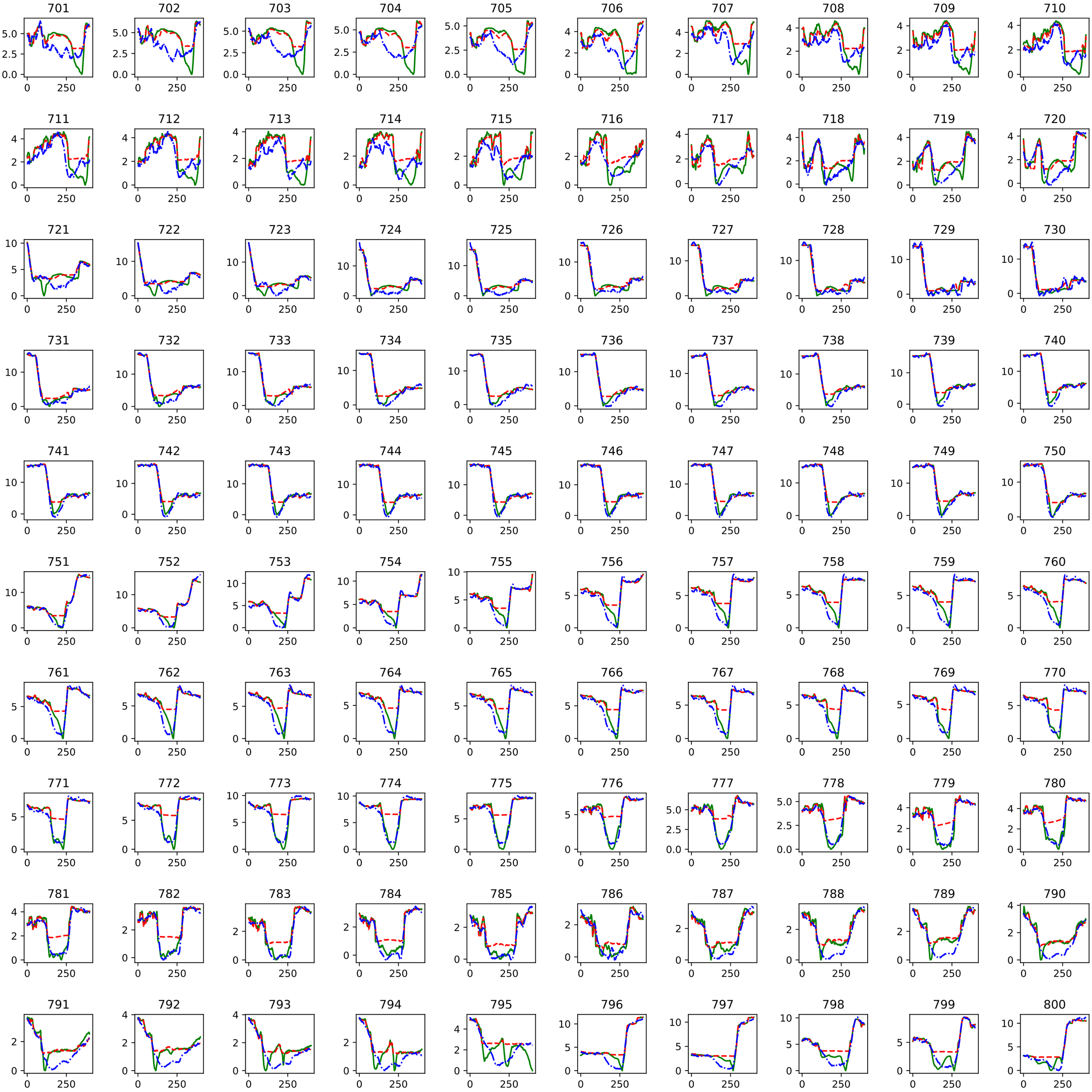
651 652 653 654 655 656 657 658 659 660

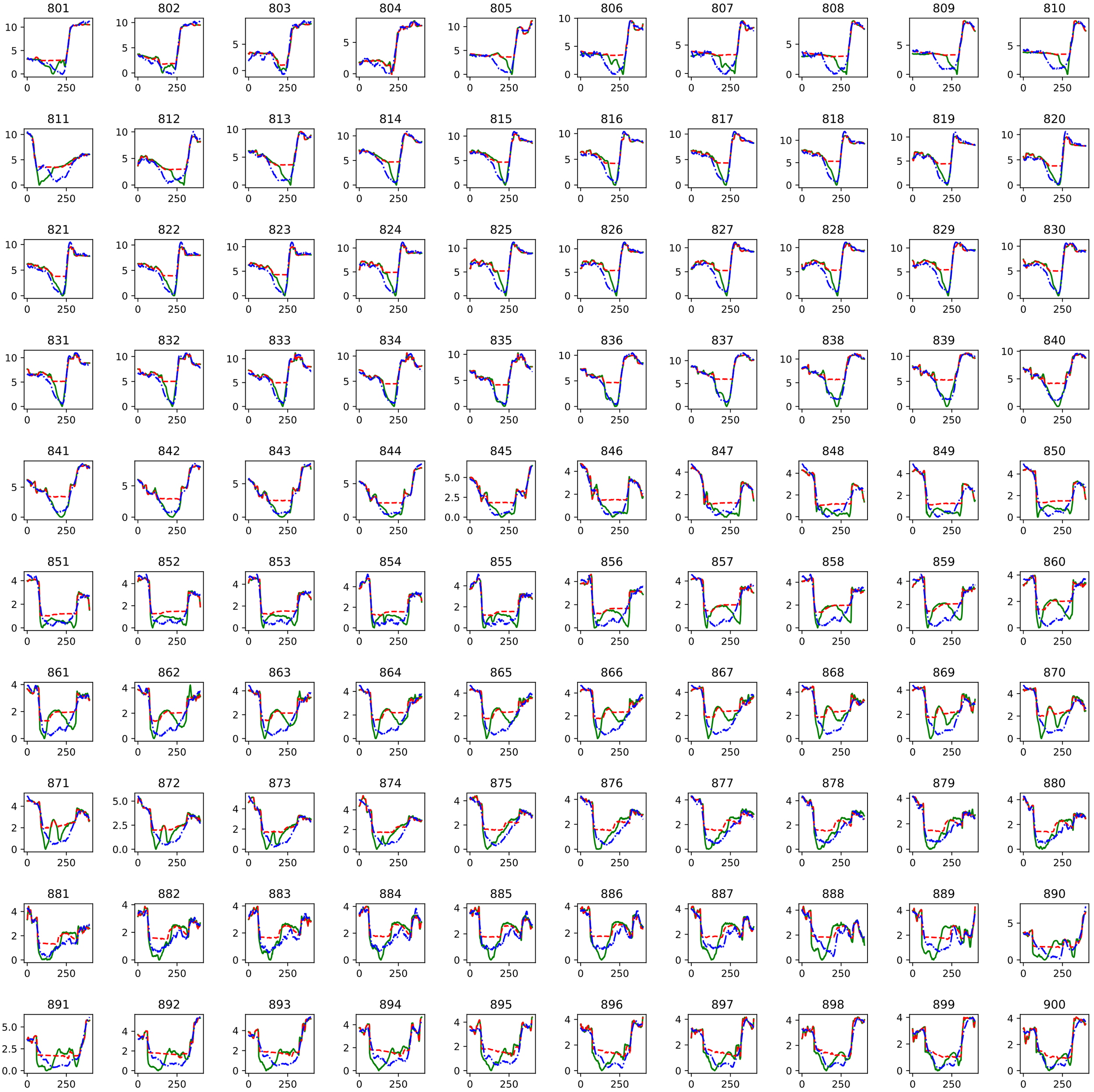
661 662 663 664 665 666 667 668 669 670

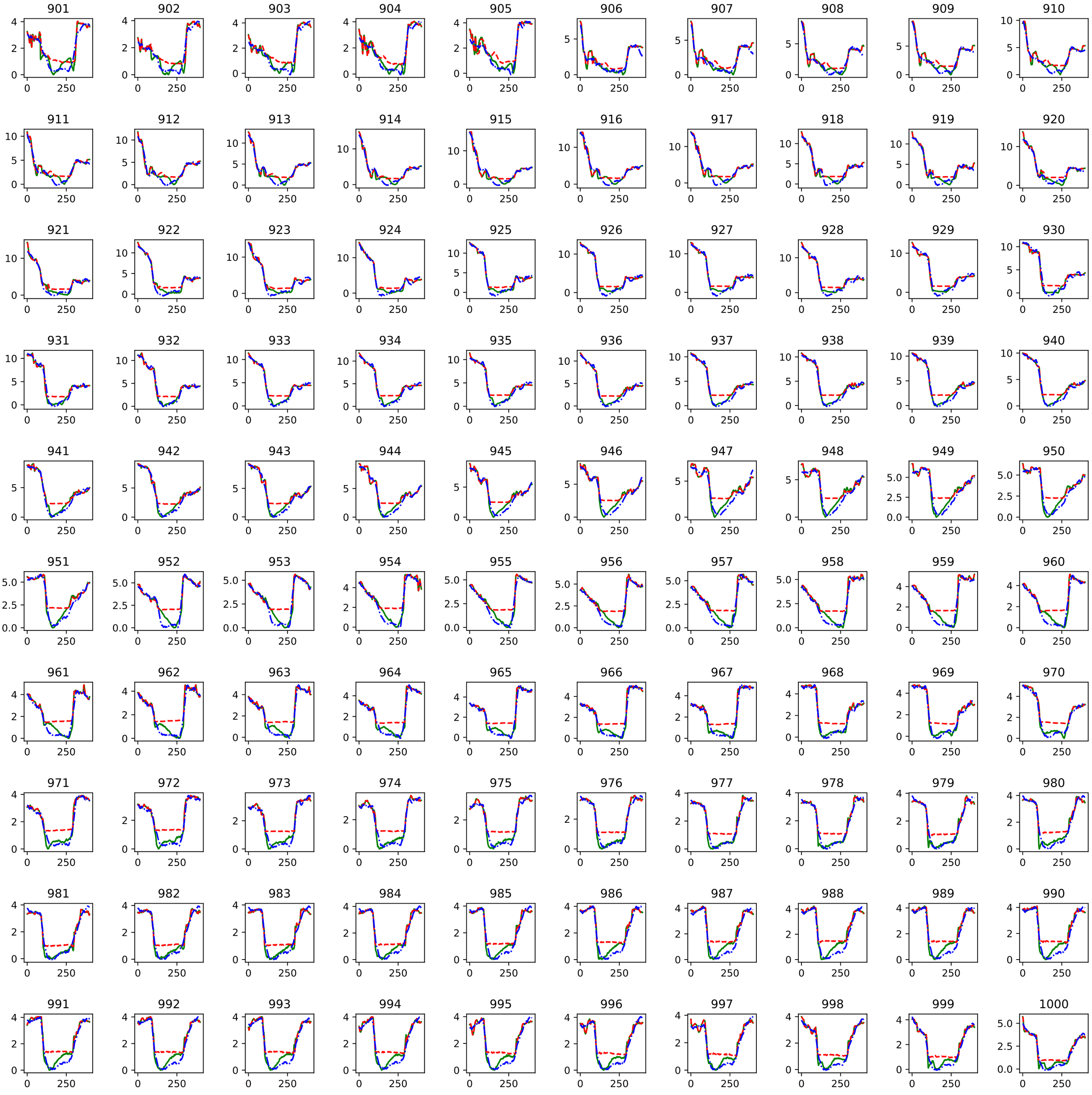
671 672 673 674 675 676 677 678 679 680

681 682 683 684 685 686 687 688 689 690

691 692 693 694 695 696 697 698 699 700


701 702 703 704 705 706 707 708 709 710
711 712 713 714 715 716 717 718 719 720
721 722 723 724 725 726 727 728 729 730
731 732 733 734 735 736 737 738 739 740
741 742 743 744 745 746 747 748 749 750
751 752 753 754 755 756 757 758 759 760
761 762 763 764 765 766 767 768 769 770
771 772 773 774 775 776 777 778 779 780
781 782 783 784 785 786 787 788 789 790
791 792 793 794 795 796 797 798 799 800


801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900


901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

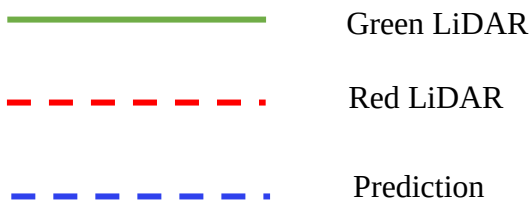
1001 1002 1003 1004 1005 1006 1007 1008 1009 1010
1011 1012 1013 1014 1015 1016 1017 1018 1019 1020
1021 1022 1023 1024 1025 1026 1027 1028 1029 1030
1031 1032 1033 1034 1035 1036 1037 1038 1039 1040
1041 1042 1043 1044 1045 1046 1047 1048 1049 1050
1051 1052 1053 1054 1055 1056 1057 1058 1059 1060
1061 1062 1063 1064 1065 1066 1067 1068 1069 1070
1071 1072 1073 1074 1075 1076 1077 1078 1079 1080
1081 1082 1083 1084 1085 1086 1087 1088 1089 1090
1091 1092 1093 1094 1095 1096 1097 1098 1099 1100

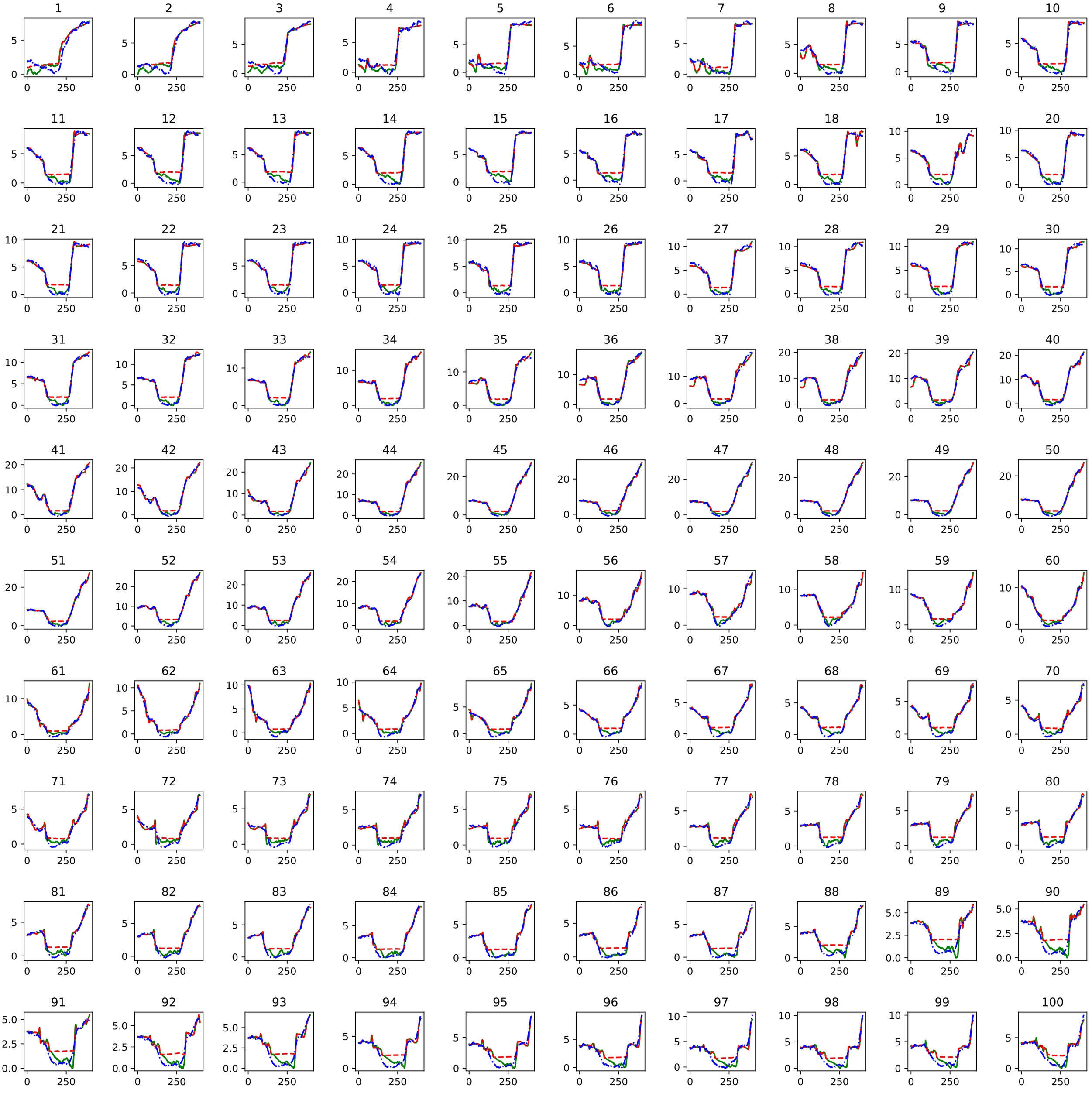
1101 1102 1103 1104 1105 1106 1107 1108 1109 1110
1111 1112 1113 1114 1115 1116 1117 1118 1119 1120
1121 1122 1123 1124 1125 1126 1127 1128 1129 1130
1131 1132 1133 1134 1135 1136 1137 1138 1139 1140
1141 1142 1143 1144 1145 1146 1147 1148 1149 1150
1151 1152 1153 1154 1155 1156 1157 1158 1159 1160
1161 1162 1163 1164 1165 1166 1167 1168 1169 1170
1171 1172 1173 1174 1175 1176 1177 1178 1179 1180
1181 1182 1183 1184 1185 1186 1187 1188 1189 1190
1191 1192 1193 1194 1195 1196 1197 1198 1199 1200

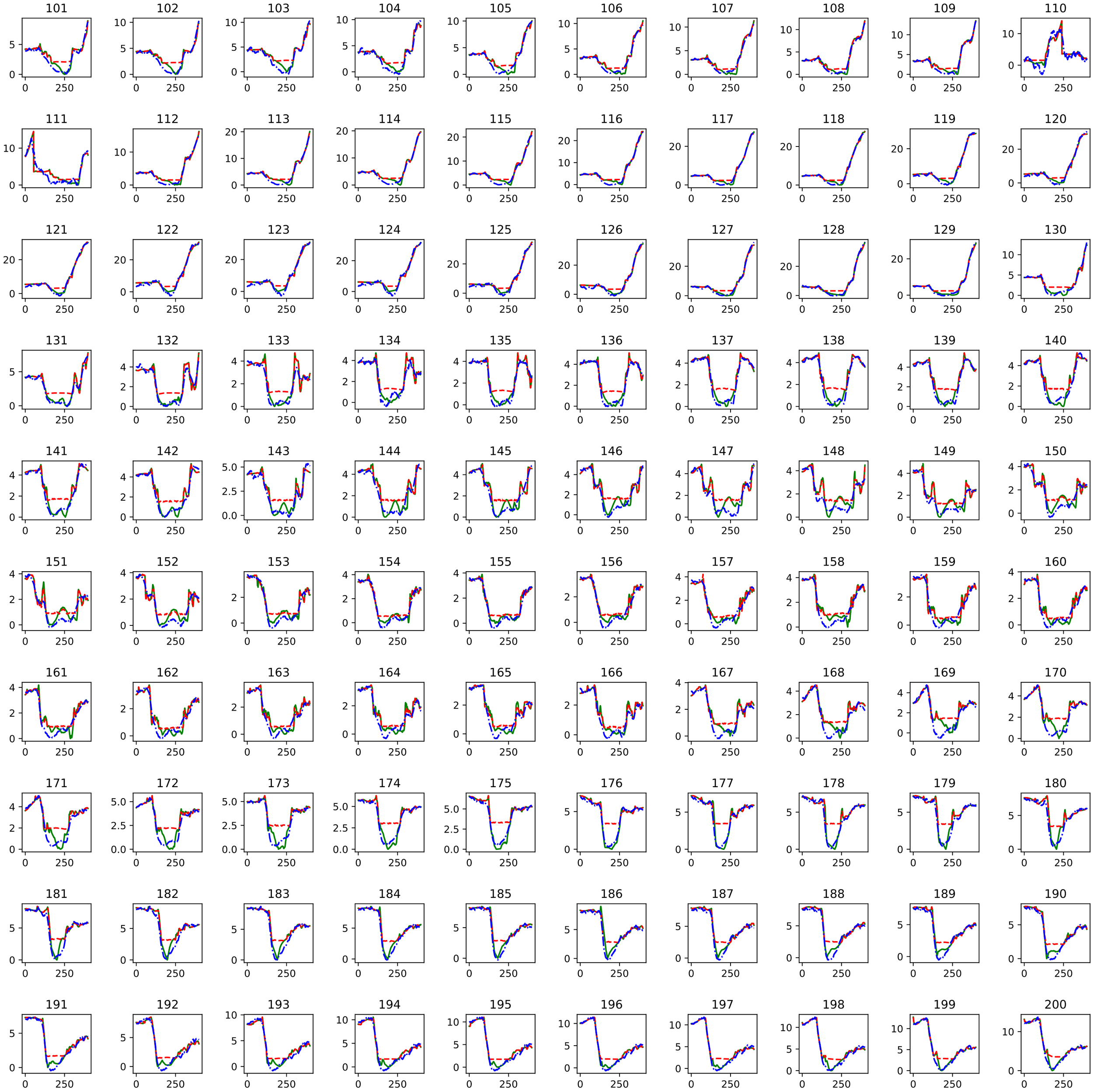
Surna ReLU

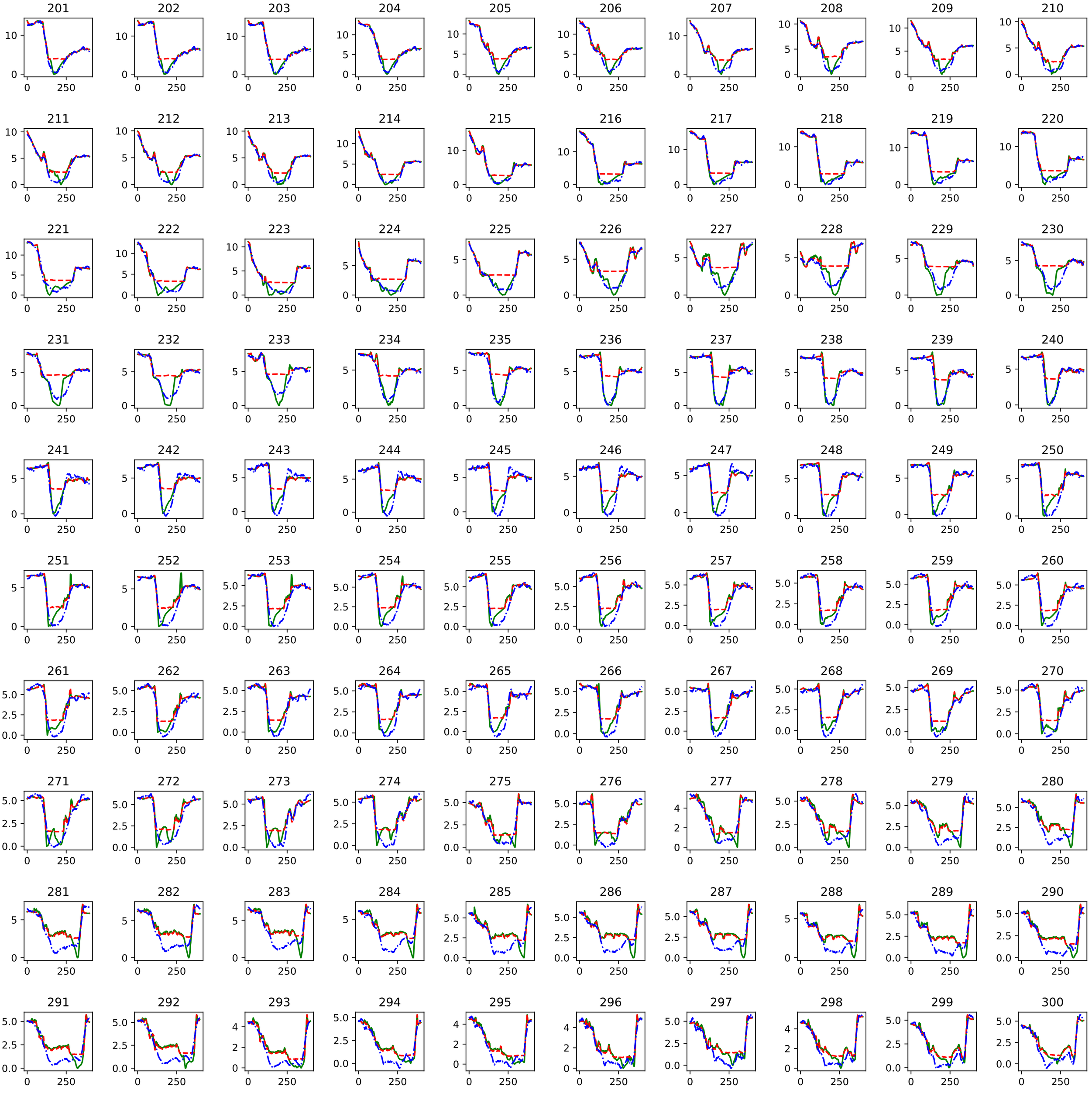
In the next pages, all the cross sections predicted for Surna are shown using ReLU activation function.

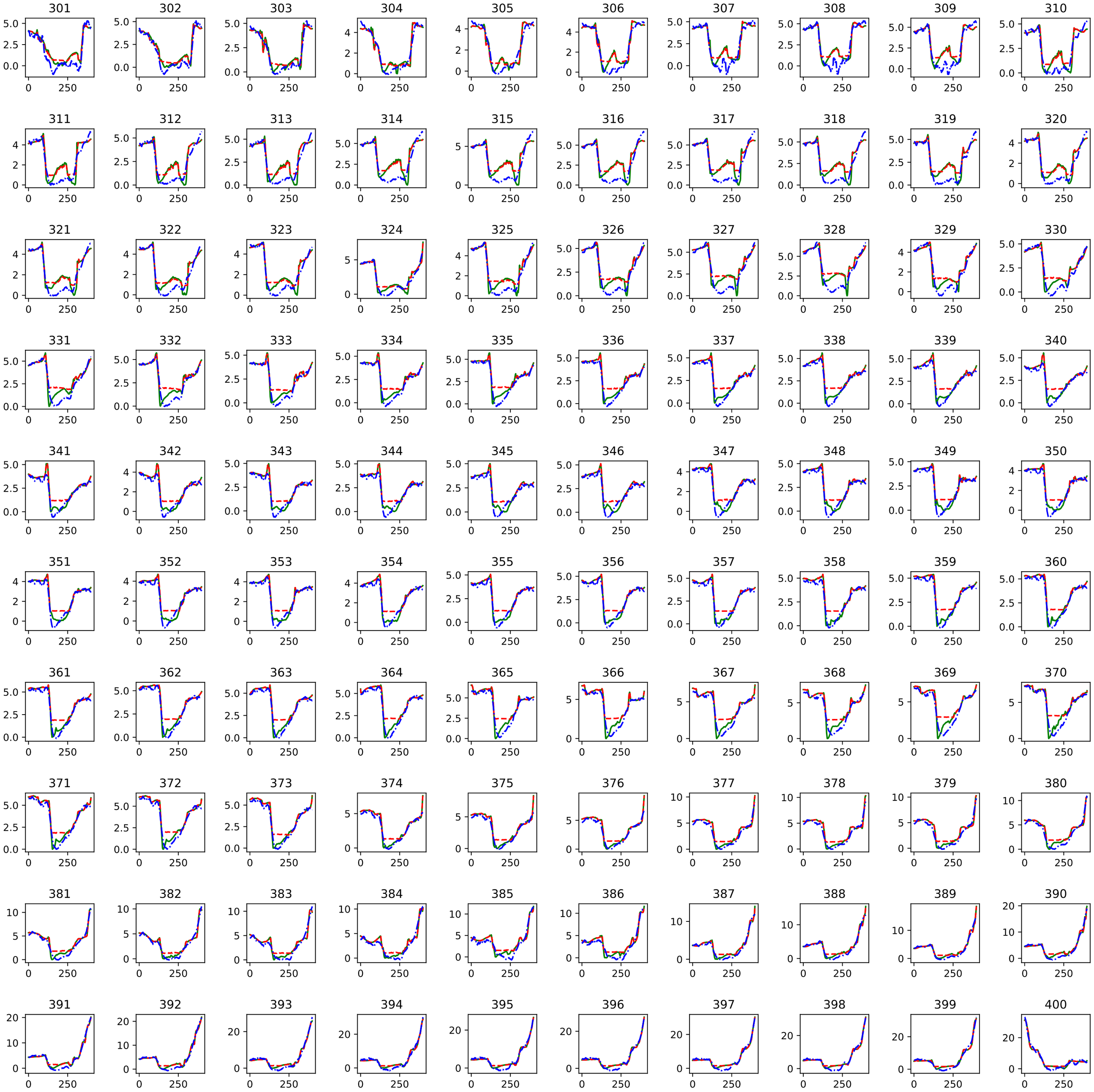
The legend for all the graphs as follow:

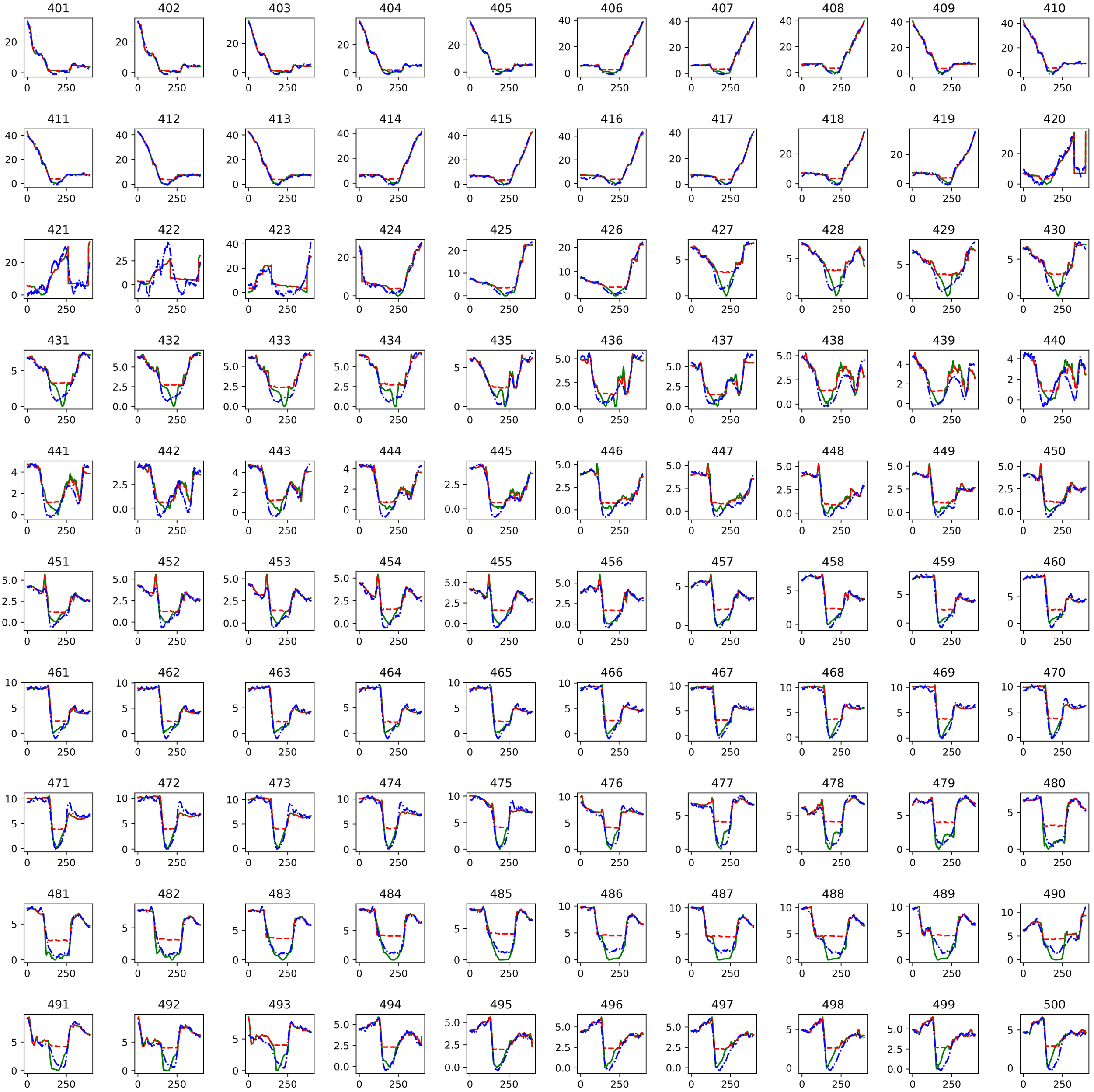



1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

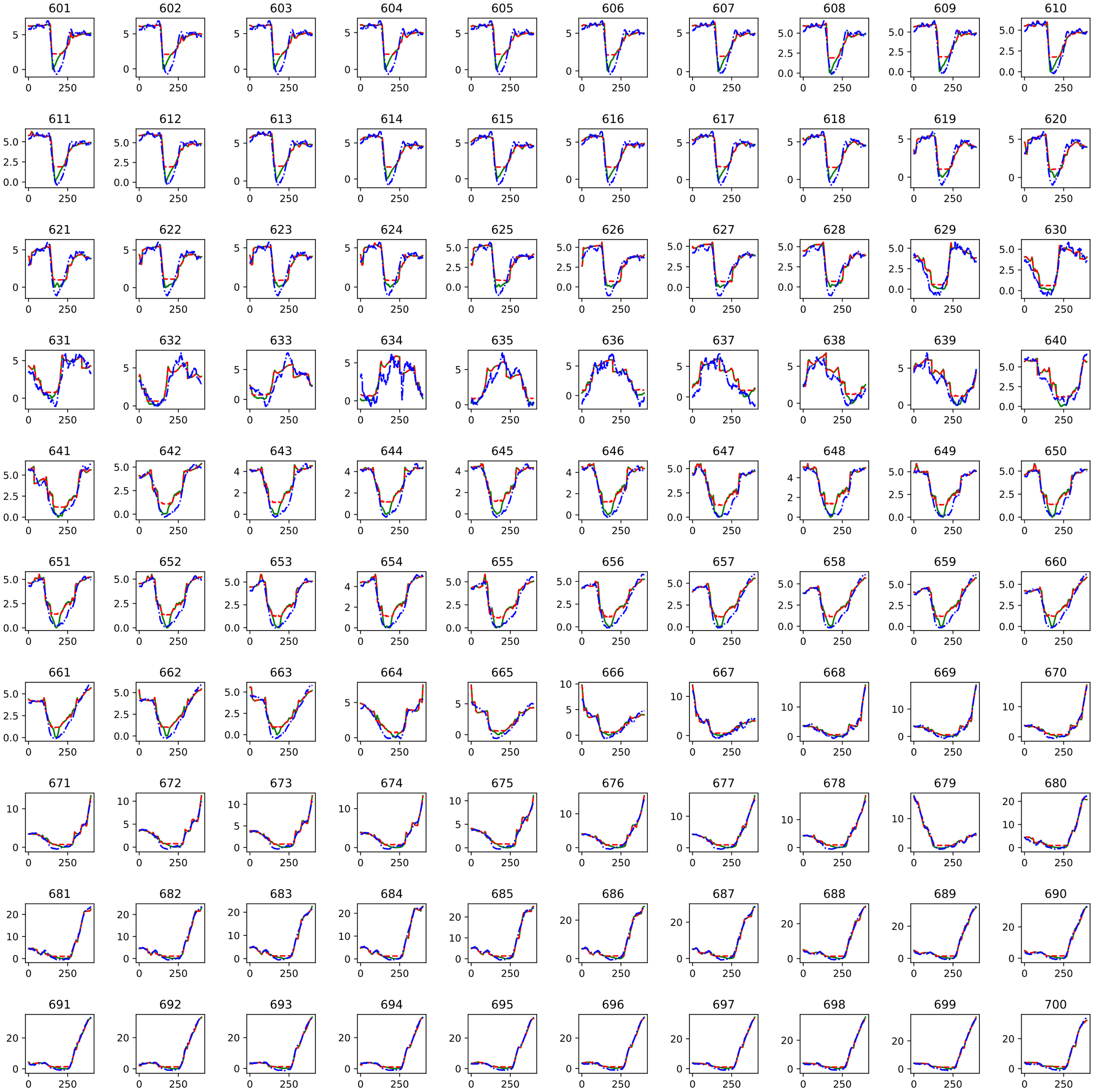

101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

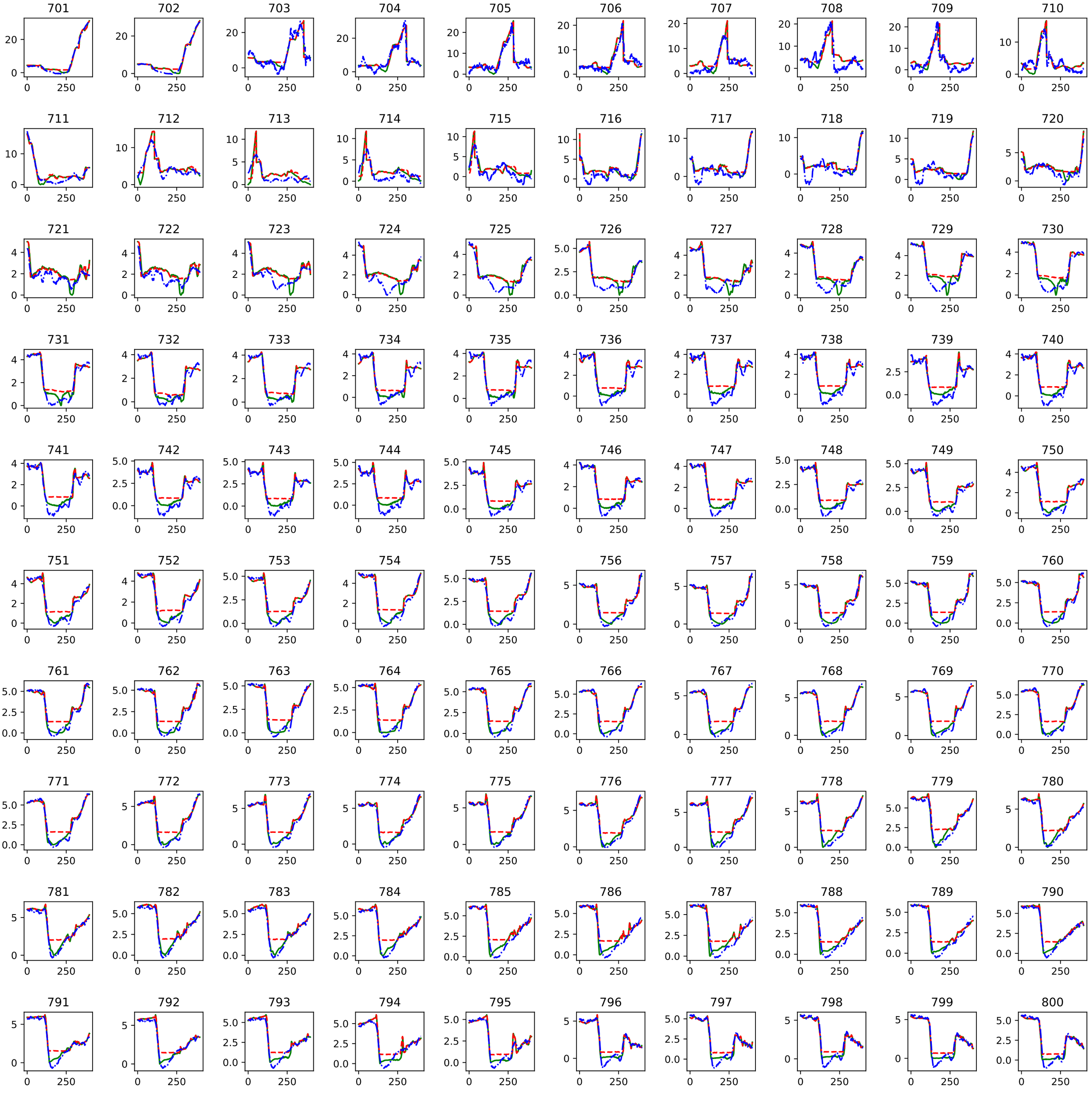

201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300

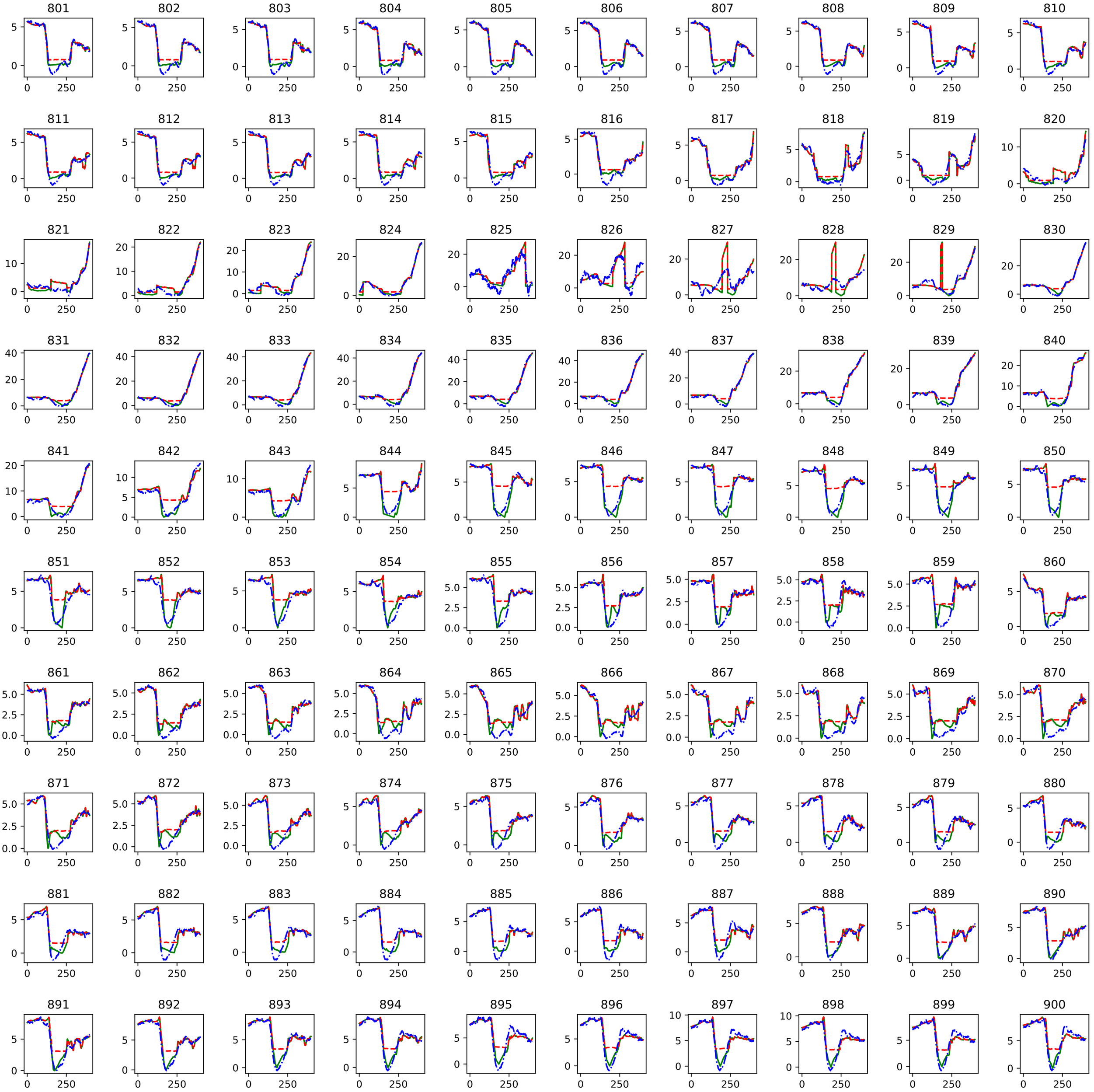

301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400

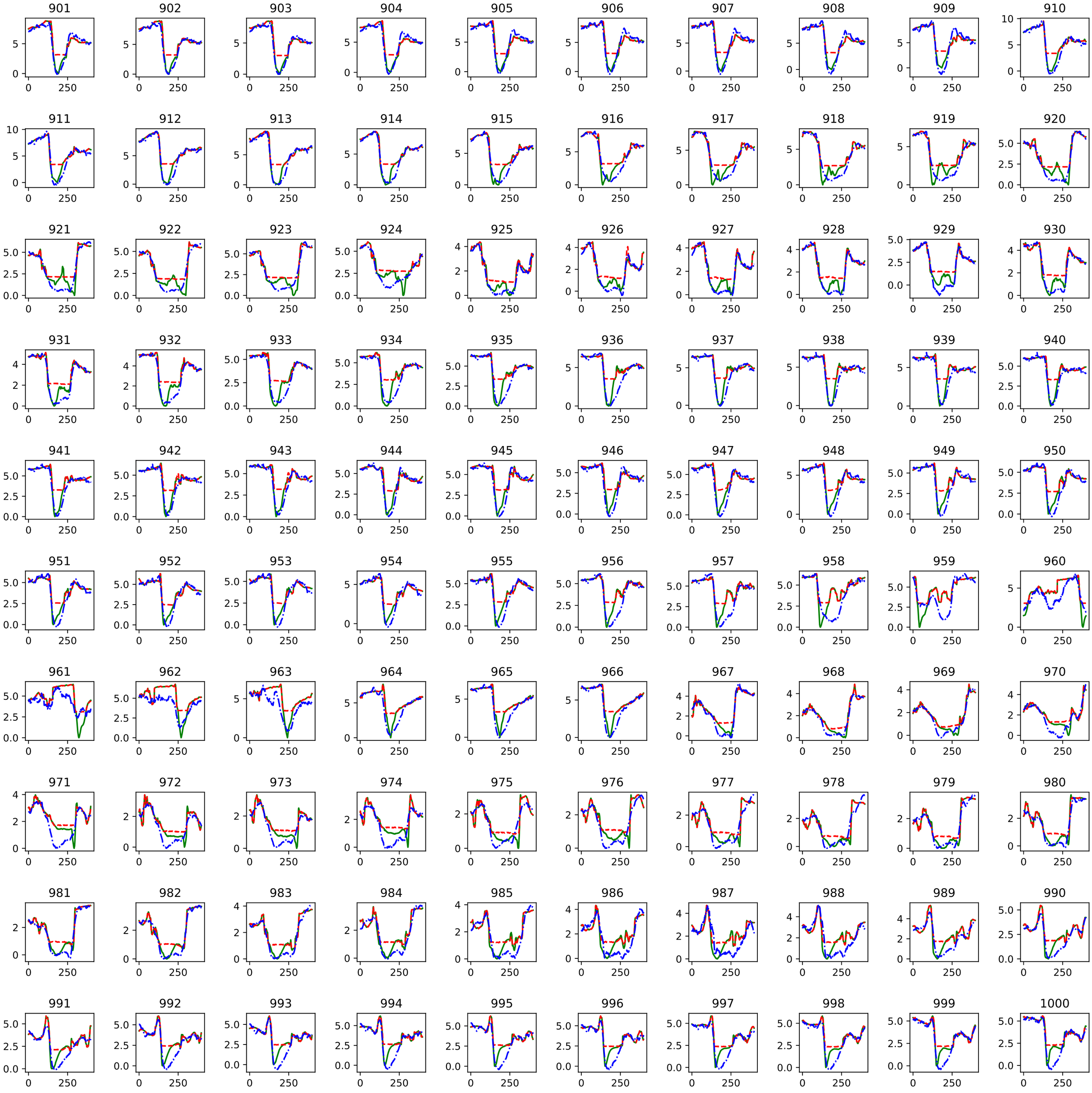

401 402 403 404 405 406 407 408 409 410
411 412 413 414 415 416 417 418 419 420
421 422 423 424 425 426 427 428 429 430
431 432 433 434 435 436 437 438 439 440
441 442 443 444 445 446 447 448 449 450
451 452 453 454 455 456 457 458 459 460
461 462 463 464 465 466 467 468 469 470
471 472 473 474 475 476 477 478 479 480
481 482 483 484 485 486 487 488 489 490
491 492 493 494 495 496 497 498 499 500


501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600


601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700

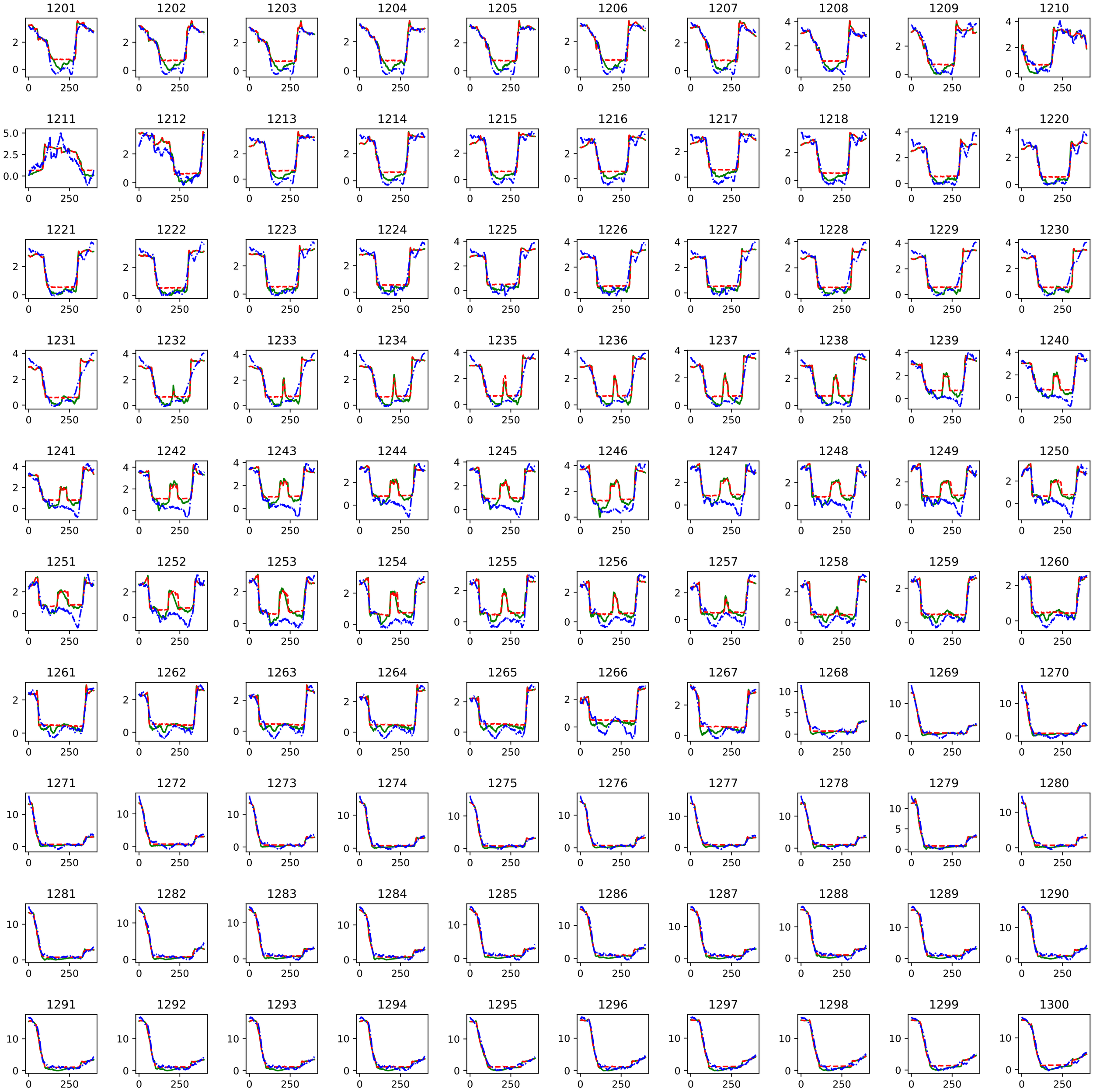

701 702 703 704 705 706 707 708 709 710
711 712 713 714 715 716 717 718 719 720
721 722 723 724 725 726 727 728 729 730
731 732 733 734 735 736 737 738 739 740
741 742 743 744 745 746 747 748 749 750
751 752 753 754 755 756 757 758 759 760
761 762 763 764 765 766 767 768 769 770
771 772 773 774 775 776 777 778 779 780
781 782 783 784 785 786 787 788 789 790
791 792 793 794 795 796 797 798 799 800

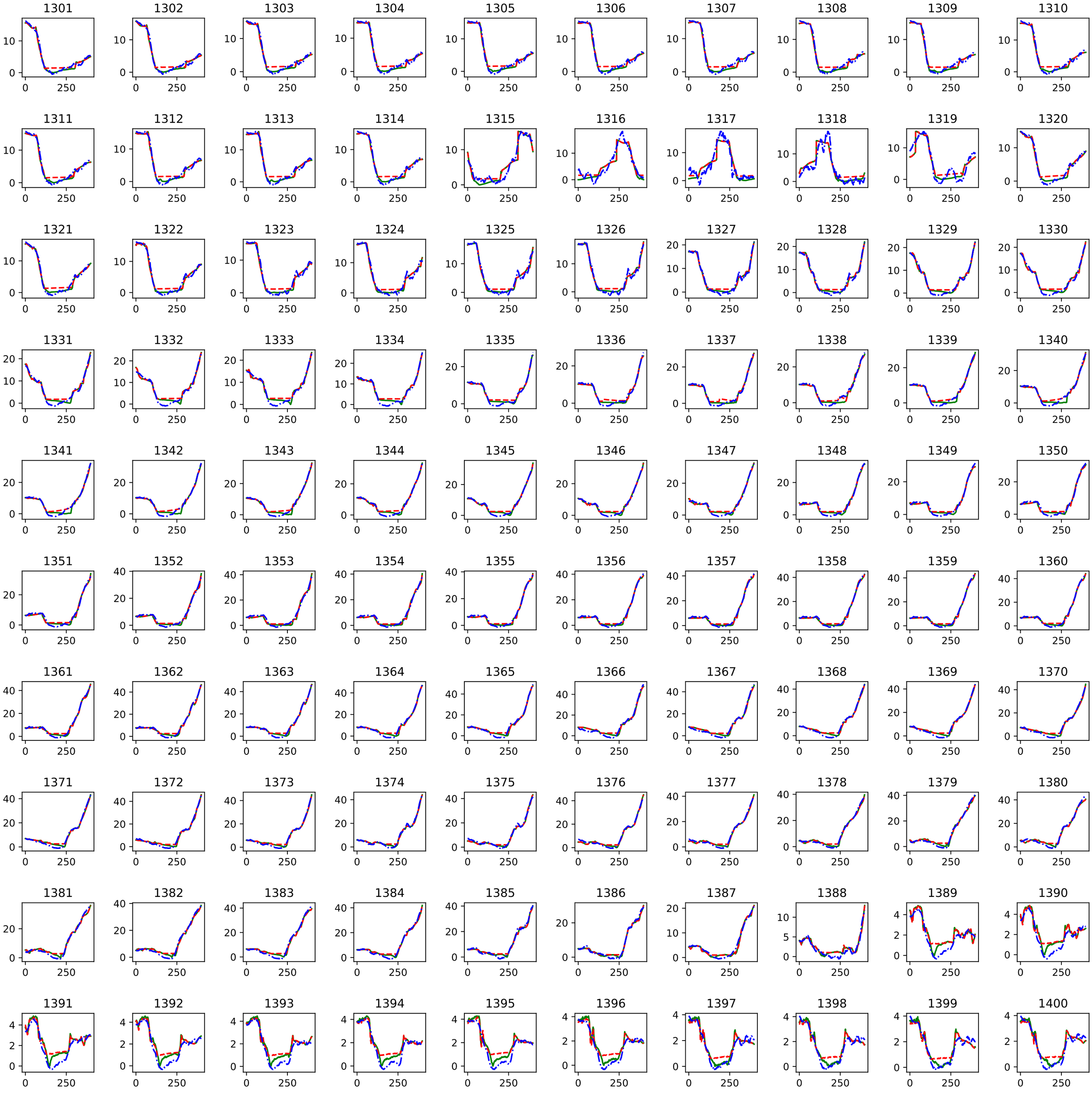

801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900

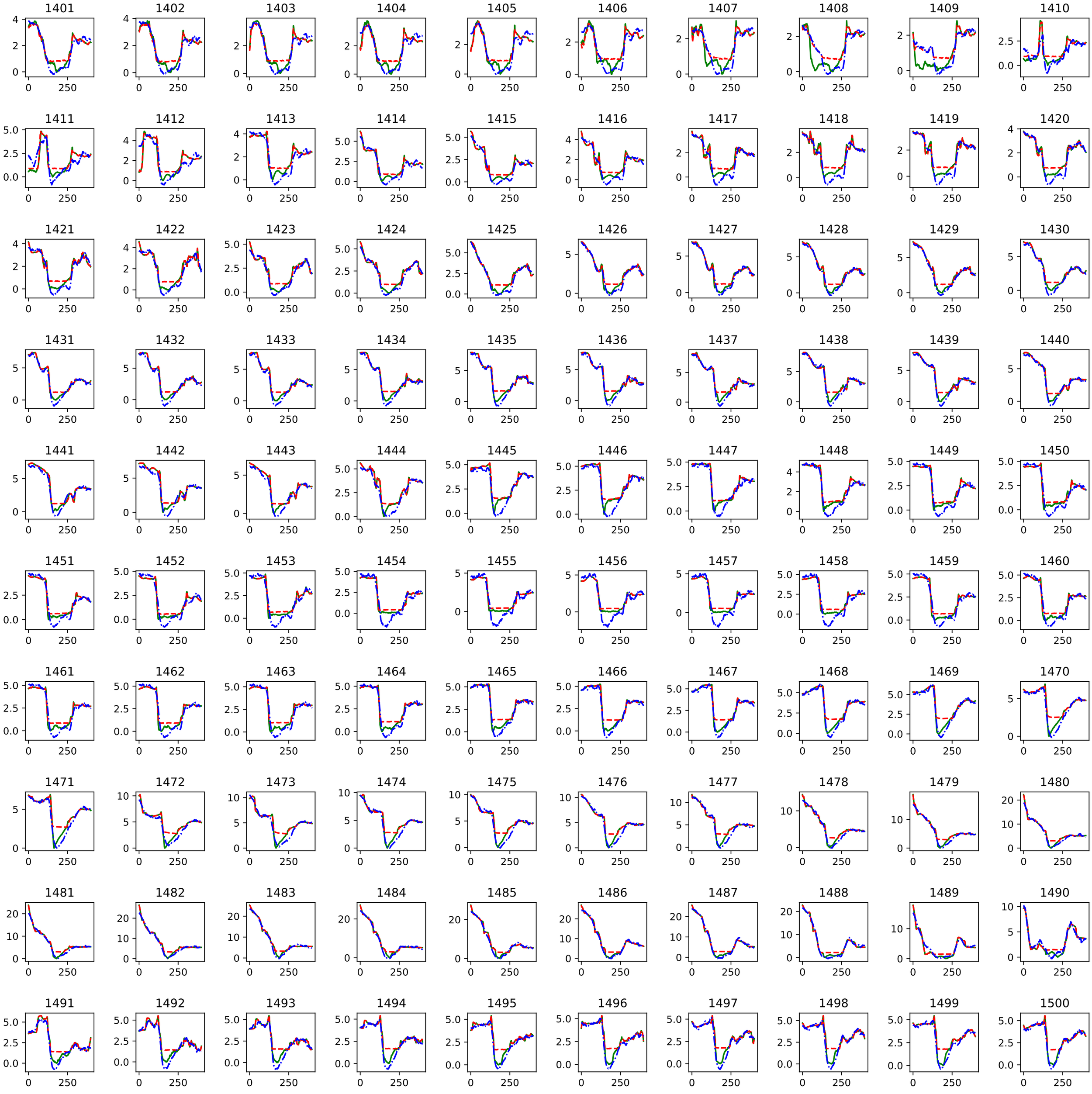

901 902 903 904 905 906 907 908 909 910
911 912 913 914 915 916 917 918 919 920
921 922 923 924 925 926 927 928 929 930
931 932 933 934 935 936 937 938 939 940
941 942 943 944 945 946 947 948 949 950
951 952 953 954 955 956 957 958 959 960
961 962 963 964 965 966 967 968 969 970
971 972 973 974 975 976 977 978 979 980
981 982 983 984 985 986 987 988 989 990
991 992 993 994 995 996 997 998 999 1000

1001 1002 1003 1004 1005 1006 1007 1008 1009 1010
1011 1012 1013 1014 1015 1016 1017 1018 1019 1020
1021 1022 1023 1024 1025 1026 1027 1028 1029 1030
1031 1032 1033 1034 1035 1036 1037 1038 1039 1040
1041 1042 1043 1044 1045 1046 1047 1048 1049 1050
1051 1052 1053 1054 1055 1056 1057 1058 1059 1060
1061 1062 1063 1064 1065 1066 1067 1068 1069 1070
1071 1072 1073 1074 1075 1076 1077 1078 1079 1080
1081 1082 1083 1084 1085 1086 1087 1088 1089 1090
1091 1092 1093 1094 1095 1096 1097 1098 1099 1100

1101 1102 1103 1104 1105 1106 1107 1108 1109 1110
1111 1112 1113 1114 1115 1116 1117 1118 1119 1120
1121 1122 1123 1124 1125 1126 1127 1128 1129 1130
1131 1132 1133 1134 1135 1136 1137 1138 1139 1140
1141 1142 1143 1144 1145 1146 1147 1148 1149 1150
1151 1152 1153 1154 1155 1156 1157 1158 1159 1160
1161 1162 1163 1164 1165 1166 1167 1168 1169 1170
1171 1172 1173 1174 1175 1176 1177 1178 1179 1180
1181 1182 1183 1184 1185 1186 1187 1188 1189 1190
1191 1192 1193 1194 1195 1196 1197 1198 1199 1200


1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300


1301 1302 1303 1304 1305 1306 1307 1308 1309 1310
1311 1312 1313 1314 1315 1316 1317 1318 1319 1320
1321 1322 1323 1324 1325 1326 1327 1328 1329 1330
1331 1332 1333 1334 1335 1336 1337 1338 1339 1340
1341 1342 1343 1344 1345 1346 1347 1348 1349 1350
1351 1352 1353 1354 1355 1356 1357 1358 1359 1360
1361 1362 1363 1364 1365 1366 1367 1368 1369 1370
1371 1372 1373 1374 1375 1376 1377 1378 1379 1380
1381 1382 1383 1384 1385 1386 1387 1388 1389 1390
1391 1392 1393 1394 1395 1396 1397 1398 1399 1400


1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500

1501 1502 1503 1504 1505 1506 1507 1508 1509 1510
1511 1512 1513 1514 1515 1516 1517 1518 1519 1520
1521 1522 1523 1524 1525 1526 1527 1528 1529 1530
1531 1532 1533 1534 1535 1536 1537 1538 1539 1540
1541 1542 1543 1544 1545 1546 1547 1548 1549 1550
1551 1552 1553 1554 1555 1556 1557 1558 1559 1560
1561 1562 1563 1564 1565 1566 1567 1568 1569 1570
1571 1572 1573 1574 1575 1576 1577 1578 1579 1580
1581 1582 1583 1584 1585 1586 1587 1588 1589 1590
1591 1592 1593 1594 1595 1596 1597 1598 1599 1600

1601 1602 1603 1604 1605 1606 1607 1608 1609 1610
1611 1612 1613 1614 1615 1616 1617 1618 1619 1620
1621 1622 1623 1624 1625 1626 1627 1628 1629 1630
1631 1632 1633 1634 1635 1636 1637 1638 1639 1640
1641 1642 1643 1644 1645 1646 1647 1648 1649 1650
1651 1652 1653 1654 1655 1656 1657 1658 1659 1660
1661 1662 1663 1664 1665 1666 1667 1668 1669 1670
1671 1672 1673 1674 1675 1676 1677 1678 1679 1680
1681 1682 1683 1684 1685 1686 1687 1688 1689 1690
1691 1692 1693 1694 1695 1696 1697 1698 1699 1700

1701 1702 1703 1704 1705 1706 1707 1708 1709 1710
1711 1712 1713 1714 1715 1716 1717 1718 1719 1720
1721 1722 1723 1724 1725 1726 1727 1728 1729 1730
1731 1732 1733 1734 1735 1736 1737 1738 1739 1740
1741 1742 1743 1744 1745 1746 1747 1748 1749 1750
1751 1752 1753 1754 1755 1756 1757 1758 1759 1760
1761 1762 1763 1764 1765 1766 1767 1768 1769 1770
1771 1772 1773 1774 1775 1776 1777 1778 1779 1780
1781 1782 1783 1784 1785 1786 1787 1788 1789 1790
1791 1792 1793 1794 1795 1796 1797 1798 1799 1800