Data Mining and Knowledge Discovery
https://doi.org/10.1007/s10618-021-00761-9

CrashNet: an encoder–decoder architecture to predict crash test outcomes

Mohamed Karim Belaid[1] · **Maximilian Rabus**[2] · **Ralf Krestel**[3]

Received: 20 September 2020 / Accepted: 30 April 2021

Abstract

Destructive car crash tests are an elaborate, time-consuming, and expensive necessity of the automotive development process. Today, finite element method (FEM) simulations are used to reduce costs by simulating car crashes computationally. We propose CrashNet, an encoder–decoder deep neural network architecture that reduces costs further and models specific outcomes of car crashes very accurately. We achieve this by formulating car crash events as time series prediction enriched with a set of scalar features. Traditional sequence-to-sequence models are usually composed of convolutional neural network (CNN) and CNN transpose layers. We propose to concatenate those with an MLP capable of learning how to inject the given scalars into the output time series. In addition, we replace the CNN transpose with 2D CNN transpose layers in order to force the model to process the hidden state of the set of scalars as one time series. The proposed CrashNet model can be trained efficiently and is able to process scalars and time series as input in order to infer the results of crash tests. CrashNet produces results faster and at a lower cost compared to destructive tests and FEM simulations. Moreover, it represents a novel approach in the car safety management domain.

Keywords Predictive models · Time series analysis · Supervised deep neural networks · Car safety management

Responsible editor: Annalisa Appice, Sergio Escalera, Jose A. Gamez, Heike Trautmann.

✉ Ralf Krestel
ralf.krestel@hpi.de

Mohamed Karim Belaid
belad01@ads.uni-passau.de

Maximilian Rabus
maximilian.rabus2@porsche.de

[1] University of Passau, Passau, Germany

[2] Porsche AG, Stuttgart, Germany

[3] Hasso Plattner Institute, University of Potsdam, Potsdam, Germany

Published online: 30 May 2021

1 Crash tests

Multi-national car manufacturers are hampered, in their race towards globalization, by the law requirements of each country. Indeed, for each new car model, a large class of crash tests with different scenarios has to be fulfilled. These tests are performed multiple times during the development process to ensure that the required safety standards are being met. Each real crash requires several weeks of organization. The development of a new car model is time-consuming and expensive as it requires hundreds of crash tests in order to fulfill all regulations and laws. A first attempt to replace crash tests and lower the costs was the introduction of crash simulations based on the finite element method (FEM simulations). Let's consider a specific crash scenario: The automotive safety assessment process embraces various front normal crash tests where a car drives against a rigid barrier at a certain speed. As an example, one is performed with a male dummy in the driver position, and one is done with a female dummy. Both tests share the same crash scenario characteristics. Thus, the car's acceleration time series can be obtained from the first real crash and used for the FEM simulations performed afterward for the female dummy. Moreover, an acceptable prediction of the car's acceleration time series could also be directly obtained using classical simulation software.

Nevertheless, the obtained estimation requires numerous working days to set up the simulation software. Then, the simulation runs for 4–33 h depending on the crash scenario, the capacity of the cluster, and the quality of the results (Böttcher et al. 2005; Vangi et al. 2018). This massive computing capacity drives the cost again to $400 or even $5000 per simulation, against $300,000 for a real crash test (Spethmann et al. 2009). The goal of this research is to decrease the cost of the safety development process by decreasing the number of necessary destructive crashes and FEM-simulations. Since the structure of the car is highly complex for physical models, we opted to apply machine learning techniques to explore new ways to model a crash event.

Deep neural network architectures are very successful in computer vision, natural language processing, and other perception tasks. In these contexts, large training datasets can be gathered consisting of unprocessed and unfiltered, raw data. In the case of supervised learning, no other data is used as input, apart from the labels, e.g., for face detection, only the raw images are used without further information. This allows us to create a large dataset without expensive annotations. Crash cars are equipped with several hundred sensors, e.g., acceleration sensors on the car structure and inside the dummies. The signals of these sensors can be understood as a time series and are used as the raw input data for CrashNet. But in contrast to large-scale perception tasks, these time series are accompanied by a set of scalar features like the car mass or the dummy type. They are used as additional information to predict crash test outcomes.

In this paper, we present CrashNet, a deep neural network that uses as input multiple scalar features and the car's acceleration time series: a 1-dimensional time series representing the deceleration of the car during a crash event. CrashNet predicts the occupant's kinematic during the crash event, by outputting a full 1-dimensional time series representing its chest acceleration. Based on this information, the injury severity of the occupant can be estimated. CrashNet can also be reshaped to output only one scalar such as e.g., the maximum chest acceleration.

2 Related work

Car safety management needs monitoring during the entire life cycle of a car. Research related to car safety starts during prototyping and continues after the production phase in order to understand the safety behavior. Research in the car safety domain can be divided into three different fields:

1. Predicting the risk of an accident based on general facts (geographic location, car speed, car model, etc.) (Chen et al. 2016).
2. Optimizing the active safety features by, e.g., improving the object detection and real-time risk evaluation modules (Sivaraman and Trivedi 2009; Szczurek et al. 2012; Sun et al. 2006) and warning distracted drivers (Trivedi et al. 2007).
3. Predicting the car deformation (Hilmann and Hänschke 2009; Grunert and Fehr 2016; Zhao et al. 2010), and the occupant behavior during a crash (Bastien et al. 2017; Iwamoto et al. 2012; Untaroiu and Adam 2012).

The first field is the most explored one by the data science community as road accident data are made available to the public by governments and agencies. The second field requires only non-destructive testing and hardware equipment (cameras, graphic cards, sensors) and research can be conducted with a small budget. However, the third field requires many destructive tests for data collection. Car manufacturers have been collecting these data for decades during the development phases of new cars. This data is confidential and therefore only car manufacturers have access to it. This explains why only a few research papers are published in this field, and, generally, there is no training data publicly available. Public research about predicting occupant behavior using data science aims either at creating crash test dummies with higher bio-fidelity or to improve computer simulations. Diverse AI techniques were applied: from classification methods (Untaroiu and Adam 2012) to reinforcement learning (Iwamoto et al. 2012). But none of them tried to predict the driver's chest acceleration time series, required by law and regulations.

2.1 Physical and mathematical models

Given enough input data and a complex model, predicting car crash outcomes is theoretically possible using machine learning. The model can learn patterns from the crash test setup, generalize over the individual data points, and predict the results of future tests. To model a crash test, two different approaches can be followed.

The first approach is based on a *physical model*. It takes into consideration the structure of the car and all physical phenomena observed during the crash (friction, torsion, etc.). Therefore, any prototype, even a completely novel type of car, can be assessed. In addition, the safety of the occupant is proven before investing in production. On the other hand, this method requires a fairly extensive description of all car components.

The second approach is supported by a *mathematical model*. The idea is to extrapolate results from different real crash tests and deduce the result for similar crash tests. This approach is applicable to similar car models. Given that commercialized

car models evolve slowly, e.g., vehicle components are used for multiple lines of cars, mathematical models are a valid approach.

Researchers have applied the mathematical approach to predict the behavior of dummies since 1995 (Melvin 1995). The working methodology relies on the principle of dimensional analysis, which states that any measure can be decomposed into combinations of mass, time, temperature, and length. The idea is to scale up the available results and infer the head injury criterion (HIC) of a similar dummy (Park and Kan 2010). Other measures can also be inferred, including the occupant chest acceleration time series. This latter is the main research focus of this paper.

2.2 Data science models

From brain signal prediction (Ullah et al. 2018) to wind speed prediction (Fukuoka et al. 2018), neural networks have been used to predict all types of time series. In general, an encoder–decoder architecture is used. The encoder handles the extraction of valuable information from the input data, whereas the decoder learns how to generate the output time series out of the obtained hidden state. In the following, we discuss the most popular layers used in the *encoder* part.

Convolutional neural network (CNN) architectures became popular since the inception of ImageNet (Krizhevsky et al. 2012). This work highlighted, in a practical example, the ability of convolutional layers to analyze the available data hierarchically and to extract local patterns automatically. In the former cited work, the extracted hidden states help classifying images. In another application (Ince et al. 2016), extracted hidden states help detecting outliers in motor signals. Detected patterns are sharpened thanks to regularization methods (Hinton et al. 2012), which allow the model to learn distinguishable patterns. Overall, CNN blocks are well-known for their exclusive property of learning patterns unrelated to the time position, which allows them to generalize on small datasets (Chollet 2017a). Although the stacked 1D CNN layer module is simple to implement, its architecture has many hyper-parameters (kernel size, number of filters, number of layers).

Multi-Resolution CNN takes advantage of different kernel sizes. Indeed, each branch is able to extract specific patterns and analyze the data at a different scale. This method has been introduced in a paper named "Attention is All You Need" (Vaswani et al. 2017). The paper promoted attention models and argued against the use of memory-based layers (RNNs).

Temporal convolutional network (TCN) is a modified version of CNN proposed by Google in 2016 (van den Oord et al. 2016). It is an audio generative model based on the PixelCNN (Van Den Oord et al. 2016). It showed impressive performance as an audio generative model. Compared to a normal CNN block with causal padding, TCN is able to connect much more input features to one output, thanks to its sparse architecture (dilated causal padding). Nevertheless, its efficiency for long-term memory applications is still debated (Wan et al. 2019; Bai 2018).

Depthwise separable convolution performs the same operation as a classical CNN layer. CNN's weights are represented as a matrix of size m by n. During a forward pass, the input is multiplied by the weight matrix. Thus, the output is calculated in

one step and m by n multiplications are performed. In contrast, depthwise separable convolution has two vectors representing its weights: a column vector of length m and a row vector of length n. This time, the forward pass is calculated in two steps. The input is multiplied successively by the two vectors. A total of $m + n$ multiplications are performed. Thus, separable convolution performs less computational operations. But intermediary results have to be saved in memory. Globally, separable convolution is significantly faster for filters larger than 4 by 4. Depthwise separable convolution has certain limitations regarding the learned filters. A separable matrix could represent a Gaussian filter or a box filter. The demonstration is based on singular value decomposition (SVD). Opposingly, hexagon filters and "donut"-type bokeh cannot be represented with separate matrices. In general, band-pass filters are not separable. To summarize, depthwise separable convolution has a limited learning capability despite the possible approximation methods. Nevertheless, it can show better results in certain applications (Wojna et al. 2019; Chollet 2017b).

Concerning the *decoder*, much less research has been conducted. Nevertheless, it is not deniable that it plays a significant role in establishing a powerful model (Wojna et al. 2019). Encoder–decoder models usually have a symmetrical architecture: CNN layers are replaced by CNN transpose layers (Dumoulin and Visin 2016) while max pooling layers are replaced by up-sampling methods.

Finally, one should consider other types of improvement of the architecture, such as activation functions and skip layers (Bishop 1995; Ripley 2007) as they have a specific added value to the model. In this paper, we experimentally evaluated and compared the various layer types for application in the crash test domain. The result of our analysis is CrashNet, a novel deep neural network architecture capable of modeling the crash test environment.

3 CrashNet

CrashNet is an encoder–decoder deep neural network architecture. It combines different layers and blocks to model the highly complex relationships between input time series data and selected scalar input features to predict the driver's chest acceleration time series.

3.1 Overall architecture

CrashNet and convolutional auto-encoder have similar topologies. Figure 1 gives an overview of the CrashNet architecture. The encoder part processes the car's acceleration time series and the scalars separately. The car's acceleration time series is encoded through two CNN blocks with max-pooling and tanh activation function. In parallel, scalar features are processed using three densely connected layers with relu activation. Thus, the output of the encoder is composed of two sets of hidden features. The first set of features is provided by the input time series. And the second results from the scalar inputs. It is now the role of the decoder to merge them. The driver's chest acceleration time series (output) is reconstructed using transpose convolutions.

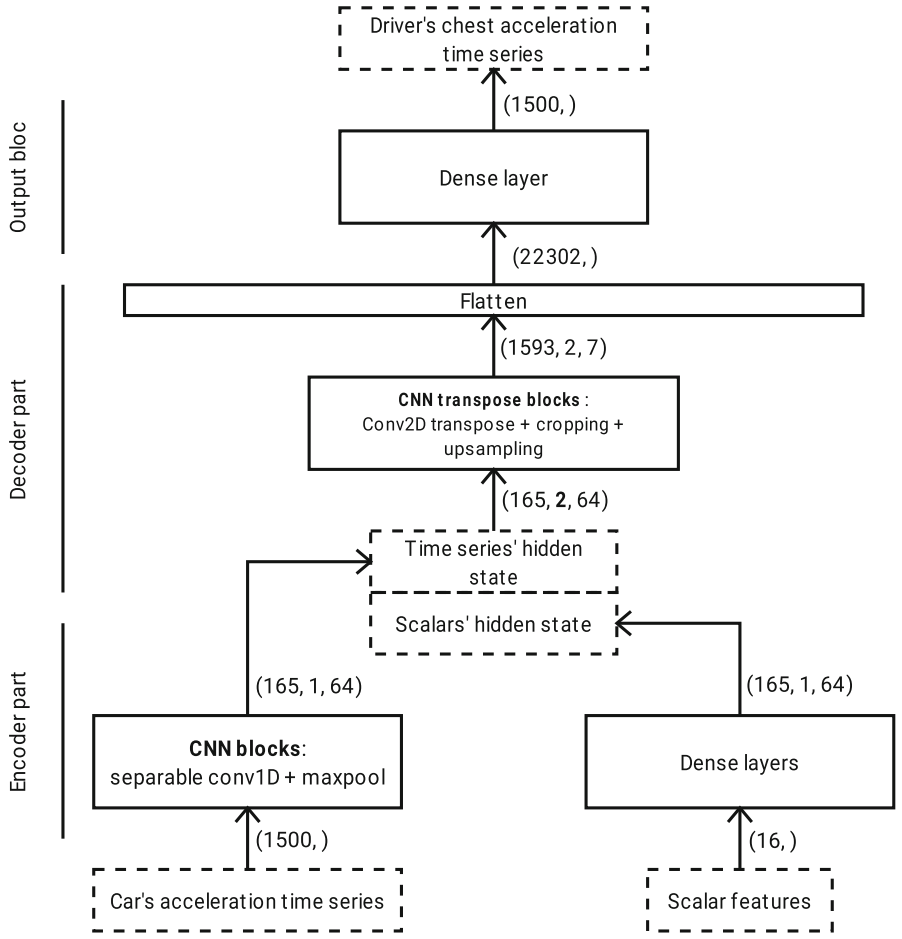


Fig. 1 CrashNet Architecture (values in brackets are output data dimensions)

We apply a 2-dimensional kernel in order to combine the latent representation of the car's acceleration time series with the scalars.

Let $k \times 2$ be the size of the rectangular kernel of the "deconvolutional" layers. The first dimension represents the time steps. The second dimension jumps from the hidden state of the car's acceleration time series to the hidden state of the scalars. The output of the deconvolutional layer is scaled up in both dimensions (Fig. 2). If we consider how the kernel is translated, we deduce that three vectors will be created out of the two input vectors: Output vector 1 results from the latent representation of the car's acceleration time series. Output vector 2 combines both car's acceleration time series and scalars. Output vector 3 is a combination of the scalar features only. Output vector 3 is, thereafter, *cropped in order to limit the influence of the scalars and to foster the merge between time series and scalars*. The cropping could not be done using conv1D transpose. Mainly, thanks to Conv2D transpose we are able to process, first, the input

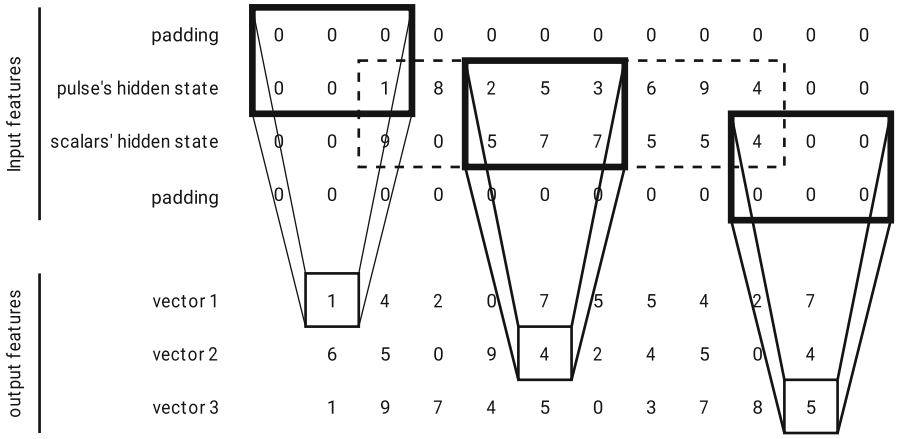


Fig. 2 Convolutional transpose operation adopted to scalar inputs

time series, then, then scalars, and the input time series with the same filters. The last layer of the block performs a bilinear upscaling through the time step dimension. The second deconvolution block performs the same three operations again. The output of the decoder, around 22000 features, is fed into one dense layer that outputs the driver's chest acceleration time series, 1500 values.

3.2 Added value of CrashNet

The following points summarize the contribution of the architecture:

- The scalar features describe the car and the driver dummy. We suppose that each scalar influences the driver's behavior (driver's chest acceleration) at a particular time step. Thus, CrashNet must have the learning capability of transforming scalars (without any time indication) into a time series. For this, the car's acceleration time series is used during the learning process.
- The use of scalar inputs, in this setting, leads to overfitting. To prevent this, CrashNet controls the flow of the scalar features through the neural network by adding multiple bottlenecks.

In the following, we describe the components and concepts in more detail. In order to integrate the scalars into the output's time series, CrashNet processes the scalars using two modules: an MLP and 2D convolutional transpose blocks. Thanks to the fully connected architecture of MLP, CrashNet is able to map each input scalar to the time segment that it affects (in the output time series). The role of the 2D deconvolution is, now, to shape the content of the scalars' hidden state. To do so, 2D deconvolution blocks first learn the filters that turn the car's acceleration time series into the driver's chest acceleration. Then, the same filter is applied to the scalars' hidden state, and, thanks to back-propagation (from CNN transpose to MLP), the filters mold the scalars' hidden state into a time series' hidden state. Mainly, CrashNet uses CNN transpose

and dense layers in order to inject each scalar exactly when it influences the output the most.

The second property of CrashNet concerns the bottleneck imposed on the scalars. Indeed, the model can overfit easily on the scalars as they represent the main difference between crash tests. However, they contain incomplete information about the car itself. The first bottleneck stands at the end of the encoder. The encoder outputs 165 time-samples by 64 features extracted from the input time series. In contrast, it outputs only 165 values after processing the scalars. This vector is duplicated 64 times to match the same shape of the car acceleration's hidden state. The second bottleneck stands in the architecture of the decoder. Features that are generated out of the scalars only are discarded. Vector 3 in Fig. 2 represents the discarded features. The cropping is applied twice since the decoder module is composed of two blocks.

3.3 Alternative architectures

In the process of developing CrashNet, various neural network architectures have been evaluated. Below is a summary of alternative architectures with competitive performances.

MLP: Fully dense layers are not able, by themselves, to capture useful patterns. Thus, all predictions are very similar.

TCN: Temporal convolutional network (TCN) is a variation of causal CNN where the connection between layers id dilated in order to connect a longer input sequence to each output node. Inspired by this approach (van den Oord et al. 2016), 5–10 CNN blocks are stacked with skip connections. The built residual network also contains dropout and batch normalization layers. After performing hyperparameter optimization, it has been found that 8–9 layers represent the best configuration. For 9 layers, each output is connected to $2^9 = 512$ input features. This implies that the dummy's acceleration is affected mainly by the last 51.2 ms. This observation is confirmed again during the analysis of the CrashNet model in Sect. 5.3.

RNN: Models based only on RNNs (LSTM or GRU) were not competitive (mean square error around 2000) compared to CrashNet. This might have been due to the long time series (1500 steps) of the input data. A better architecture combines RNN and CNN layers. We have found that the best combination is to alternate them. Or one could obtain this architecture by replacing the MaxPooling layers in CrashNet with RNN. This makes the model fully trainable. Despite the complexity of this solution, it showed the same mean square error (MSE) as CrashNet. But since training RNNs takes much more time than training CrashNet, we opted for architectures without recurrent layers.

To summarize, we propose a unique neural network architecture to simulate car crashes. CrashNet's architecture can be generalized to predict and generate any time series using scalars and time series as input. The encoder and decoder modules learn how to turn scalars into a time series through wrestling between the MLP and the filters of CNN transpose. Overfitting due to the scalar is reduced thanks to the bottlenecks, sprinkled throughout the layers. Finally, CrashNet has the ability to learn how to predict the driver's chest acceleration with respect to causality without weight constraints.

4 Evaluation

In order to evaluate the performance of CrashNet, we compared the prediction results with a baseline and with the noise threshold, which is a natural lower bound for the obtainable MSE. To the best of our knowledge, there is currently no state-of-the-art model that is capable of predicting the driver's chest acceleration time series. The baseline outputs the average time series over the training set. The baseline is not a model. Thus, it doesn't present a validation score.

4.1 Dataset

In this section, we describe the dataset used in this study. The input variables consist of 16 scalars and the car's acceleration time series, a array of 1500 values representing the car acceleration during the first 150 ms of the crash. The output of the model is the full time series representing the driver's chest acceleration. It is also composed of 1500 values (one value every 0.1 ms). By law, the occupant's maximum chest acceleration must stay below a certain threshold. For this reason, we are interested in predicting it. For both time series, $t = 0$ represents the first instant of interaction between the barrier and the car. The input and output time series are processed with a low pass filter, known as Channel Frequency Class Filter, which is a standard in car safety management (Grenke 2002). The stratified split is made according to the car model since a higher correlation between these car's acceleration time series has been noted.

During a crash test, some accelerometers might fail. These failures result in a discontinued time series. In our dataset, we discarded three incomplete time series resulting in a dataset composed of 450 destructive crashes plus 52 computer simulations. The 52 computer simulations represent very similar crash scenarios. In this research, we leverage simulation data to enable the model to learn how minor changes in the input can affect the driver's chest acceleration. We train models with and without simulation data to confirm the intended effect of this augmentation. We found out that the test error decreased by 3% and the quality of the predictions improved significantly. Since the dataset is relatively small, the choice of the test set has a high influence on the results. To mitigate this effect, we performed three runs of 5-fold cross-validation, which yields more robust average scores compared to one run of 5-fold-cross-validation. The 52 computer simulations constitute the training set ($450 \times 4/5 + 52 = 412$ data-points in train and validation set). The remaining 90 data points constitute the test set. The validation set represents 10% random samples of the training set ($412 \times 0.1 = 41$ data-points) and is used to monitor the learning process. Further, we use early stopping and Bayesian optimization for hyperparameter optimization.

4.1.1 Input features describing the car

In total we identified 16 relevant input scalars. One part is extracted from the crash setup, while the rest is derived from the car's acceleration time series. The following paragraphs contain a description and an exact definition of the used features. If the feature's name starts with "car", then the feature applies to any dummy independently

on its position inside the car (driver or passenger). Since we limit this work to the driver, we do not consider features related to the passenger.

car::Sliding_Mean ($SM_{25\text{ms}}$): The sliding mean describes the maximum average car deceleration within a defined time interval. For this purpose, a time window of 25 ms is used as a criterion to classify the hardness of a car's acceleration time series.
car::Initial_Speed (v_0): This is the speed of the car at the time of impact ($t = 0$). Crash tests are performed at predefined speeds: at 26, 32, or 40 km/h with unbelted dummies, and at 50 or 56 km/h with belted dummies.
car::Rebound_Speed (v_r): This is the maximum reverse speed of the car after hitting the wall. $v_r = min\left(\int [a]_{CFC180}\, dt + v_0\right)$ with a being the car's acceleration time series and v_0 the initial speed of the car.
car::Kinetic_Energy_at_t0 (E_{kin}): $E_{kin} = \frac{1}{2} \cdot m_{car} \cdot v_0^2$ with m_{car} being the car mass.
car::Acceleration_Average and car::Acceleration_Max: These are the average and max car acceleration values. In practice, they are known to correlate well with the maximum of the driver's chest acceleration ($r = 0.46$ and 0.59, respectively).
car::Acceleration_Over_3ms_Max: This is the highest acceleration recorded for a contiguous period of at least 3 ms.
car::Group: This is the model of the vehicle. This feature is used for the stratified split and is one-hot encoded into the following features: car::Group_is_Limousine, car::Group_is_Small_Sports_Car, car::Group_is_Big_Sports_Car, car::Group_is_SUV.

4.1.2 Input features describing the driver dummy

Different crash test dummies are used to assess the severity of injuries in a crash. Each dummy type is equipped with specific sensors, e.g., acceleration sensors in the dummy's chest. Since the dataset is limited to full-frontal crash tests, the HYBRID-III family is the most common type of dummy used.

driver::is_H3_Dummy	This is set to one if the driver is a HYBRID-III 50$^{\text{th}}$ percentile male dummy and it is set to zero for a HYBRID-III 5$^{\text{th}}$ percentile female dummy.
driver::Seatbelt_Used	The usage of the seat belt is not mandatory in certain countries. For this reason, car manufacturers must perform crashes with belted and unbelted dummies (Hollowell et al. 1999).
driver::ROLCp	This is the real occupant load criterion predicted (ROLCp) (Rabus 2019), which is a recent structural criterion developed in 2019. ROLCp is an estimation of the linear deceleration experienced by the occupant, with a perfect restraint system. To understand how to calculate the ROLCp, let us understand the two phases of the deceleration of the dummy:

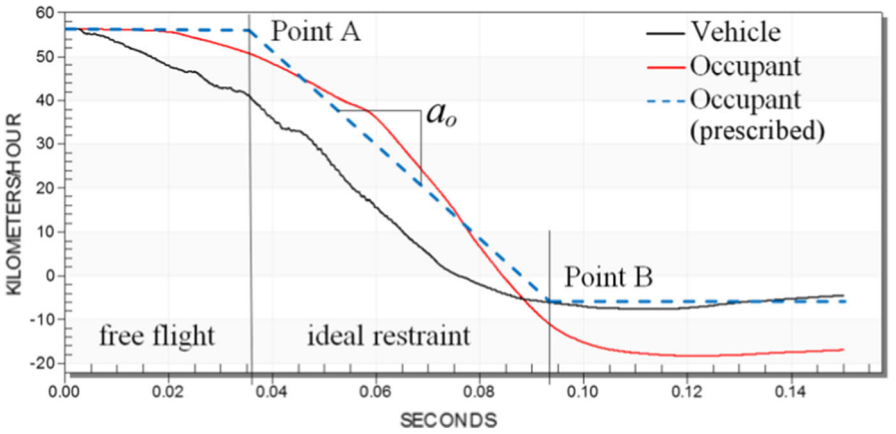


Fig. 3 Example of a NCAP test (Park and Kan 2010)

– At the beginning of the crash, the dummy is almost not connected to the car. One of the passive safety features that make the dummy decelerate is the friction with the seat (Anti-submarine seat design). Thus, the dummy keeps traveling to the front until that it is restrained by the seat belt or until it interacts with the airbag. During this first phase, the dummy does not decelerate considerably compared to the car.
– During the second phase, the dummy is connected to the car thanks to the seat belt and the airbags. Thus, the dummy's relative speed to the car is completely decreased. During this phase, the limits of a human body could be reached easily. The two phases are delimited by three time-frames $t = 0, t_A$ and t_B (see Fig. 3). The ROLCp is the absolute value of the slope connecting the two points, the car speed at t_A and at t_B. Physically speaking, the slope represents a perfect restraint system where the occupant's acceleration is constant (a_0). The ROLC is the theoretically minimum acceleration of the dummy while it is restrained inside the occupant's compartment. ROLC values are measured for a fixed distance traveled by the occupant during its deceleration. Points A and B are chosen dynamically according to the car model. They are picked without relying on the dummy's chest acceleration time series. For this reason, it is called ROLC predicted. Note that ROLCp is one of the most important scalar features as it shows the highest correlation with the target. Indeed, Pearson's r is equal to 0.83 between ROLCp and the maximum of the chest acceleration. The second-best scalar feature is driver::OLC++ and it has a correlation of only 0.76.

  driver::OLC: OLC is a specific case of ROLCp. Instead of using dynamic parameters like in ROLCp that depends on the specific car configuration, the OLC uses fixed parameters. For further detail about the choice of points A and B please have a look at (Park and Kan 2010).
  driver::OLC++: is a combination of three useful structural criteria. $OLC++ = \alpha_1 OLC + \frac{\alpha_2}{(t)_{v=0}} + \alpha_3 SM_{25\text{ms}}$ with $\alpha_1, \alpha_2, \alpha_3$ being coefficients, $(t)_{v=0}$ the time to zero velocity, $SM_{25\text{ms}}$ the car::Sliding_Mean, and $\Delta t = 25$ ms.

Table 1 Average mean squared error (MSE) ± standard error (SE)

	Train			Validation			Test		
	MSE		SE	MSE		SE	MSE		SE
Baseline	3896	±	12				3910	±	94
MLP	2413	±	13	2332	±	23	2407	±	51
TCN	1462	±	14	1903	±	34	1934	±	35
Seq2seq	1268	±	12	1431	±	24	1416	±	35
RNN + CNN	758	±	7	924	±	23	922	±	21
CrashNet	706	±	10	881	±	22	875	±	12
Noise threshold	585			585			585		

4.1.3 Tolerated level of error

Crash test datasets exhibit a significant degree of noise. The robustness noise is a well-defined and well-known type of noise in the automotive safety field (Bohlien 2016; Kang 2005; Will et al. 2006). Let us consider two comparable crashes. The obtained car's acceleration time series (input of the model) are not identical. Of course, the same holds for the observed dummy's chest accelerations. This is due to diverse tolerances in the production and the setup of the test. This noise has been quantified and in 95% of the cases, the MSE is below 585.

4.2 Results

To compare CrashNet's predictions to real crash tests, we make use of the MSE metric. A direct comparison between CrashNet and FEM simulations using MSE scores is unfortunately not possible given the following reason: In general, the accuracy of FEM simulation results varies in the early stages of the development process and is constantly improved due to validation steps, e.g., after performing the first physical crash tests. This renders a fair comparison of MSE scores between CrashNet and FEM simulations impossible. Nevertheless, the comparison between CrashNet and real crash tests is the most important indicator of the quality of the model.

4.2.1 Overall results

Aggregated results can be found in Table 1. For each model and data set, we provide the average over 15 runs plus-minus the standard error of the mean. CrashNet is outperforming the baseline by a very large margin. Its MSE value is close to the noise threshold and the MSE values are stable. For a qualitative evaluation of the generated predictions, we refer to the annex file which contains more example predictions. Besides, CrashNet can extract useful patterns, which we will investigate in the following subsections.

Moreover, the training is much faster than any former method. One real, destructive crash test requires 3–5 weeks of preparations (delivery of the prototype, dummy

Table 2 Overall comparison of the passive safety assessment tools

	Execution time	Availability before the start of production	Costs
Crash test	$\sim$ 5 weeks	12 months	$> 100,000$ €
FEM simulation	$\sim$ 30 h	24 months	> 300 €
CrashNet	$\sim$ 1 s	30 months	$\ll 1$ €

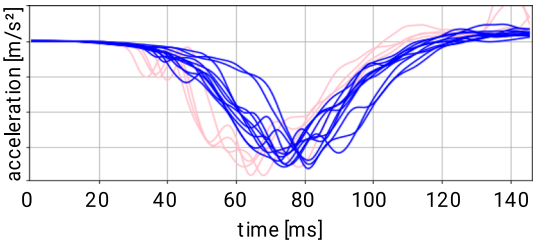
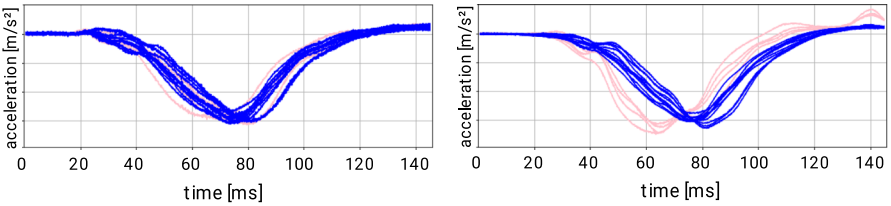


Fig. 4 Ground truth output time series of 15 crashes of the same car model[1]



(a) Prediction using Seq2seq **(b)** Prediction using CrashNet

Fig. 5 Comparing predictions using seq2seq and CrashNet model[1]

inspection, etc.) plus one day to perform the crash and process the data. On the other hand, FEM simulations would require a few days to set up the car model (dimension and stiffness of each component) plus 4–33 h of computation on a high-end machine. Finally, CrashNet is substantially less demanding: training and testing takes $\sim$ 14 mi, using one CPU and inference time is roughly one second. See Table 2 for an overview.

4.2.2 Importance of scalar features

We demonstrate the importance of the scalar features by comparing CrashNet to a sequence-to-sequence (seq2seq) model. The seq2seq model shares the same parameters as CrashNet, but it has only the car's acceleration time series as input. Table 1 summarizes the performance of both models. A significant drop in the MSE is observed in the train and test. Without the scalars, the model is clearly underfitting.

The improvement can also be qualitatively observed. In Figs. 4 and 5, a set of crashes sampled from the training set are compared. All selected crashes are performed with the same car model and the same initial speed. All dummies are unbelted. The only notable difference is the dummy type. 4 crash tests employ female dummies (pink time series) and 11 crashes are conducted with male dummies (blue time series).

Figure 4 shows the expected outputs.[1] The main difference is that the chest acceleration of female dummies reaches the lowest value around 15 ms earlier compared to male dummies because female dummies are placed closer to the steering wheel. Figure 5a illustrates the predictions of the seq2seq model. The model is unable to differentiate the reaction of male and female dummies. CrashNet's prediction is more accurate (Fig. 5b). Male and female chest acceleration time series are visually separable as they should be. Using scalars to generate a time series is challenging since already a small number of them may lead to overfitting. But with the CrashNet architecture, the influence of the scalars on the output is limited and thus the model does not overfit.

5 Discussion

This section is dedicated to the patterns learned by CrashNet: The first subsection compares the importance of each scalar feature. The second subsection demonstrates the time dependency between scalars and the output time series. The last subsection explains the causality between input and output time series.

5.1 Which scalar features are most important?

In the following paragraphs, we estimate the importance of each input scalar within the trained model using the *mean decrease accuracy* algorithm, also known as the *permutation importance* algorithm. It was initially introduced by Breiman in 2001 (Breiman 2001). Afterward, a model-agnostic version was developed in 2018 and explained in a paper named "All models are wrong, but many are useful" (Fisher et al. 2019). The algorithm measures the decrease in the score when a feature is not available. Technically, the model requires an input value for the tested feature. The tested feature cannot be discarded. The trick is to feed in random noise, i.e., no useful information. To not break the model, the generated noise is drawn from the same distribution as the original set of values. In practice, this means shuffling the original feature values. Thus, the distribution remains the same, and the model is less likely to malfunction. In this way, permutation importance can be measured. For a better understanding of Fig. 6, let us consider the dummy type feature named *driver::is_H3_Dummy*. The values are binary: male dummy or female dummy. After shuffling the dummy type values, the test MSE (eventually) increases by a certain value (here 88%). Thus, it is estimated that the importance of this feature is prorated to the increase. The permutation importance algorithm confirms the importance of binary features, a task that was not feasible with *Pearson's r*.

The *driver::is_H3_Dummy* feature obtained the highest score, which means that dummies behave differently as seen in the subsection above. And this confirms again the need to perform crashes with male and female dummies.

[1] This research was conducted in collaboration with Porsche AG. The provided data has been anonymized: sensitive data—such as measurements and the associated car model—has been masked.

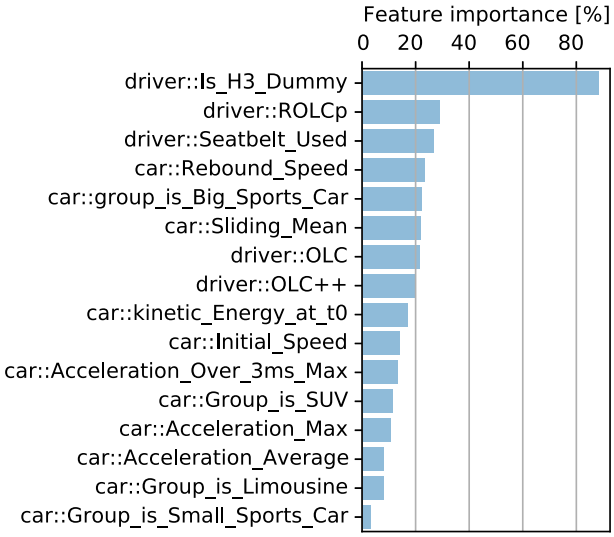


Fig. 6 Importance of scalar features for prediction outcome

The driver's ROLCp feature has a high correlation with the target variable and is among the top features selected by the trained model. This confirms earlier findings (Rabus 2019) about the usefulness of ROLCp also as a standalone feature.

The *driver::Seatbelt_Used* feature is a binary feature with a uniform distribution in the dataset. When the feature importance algorithm replaces it with a random value, this random value still has a 50% chance to be correct. Therefore we expect that the total generated error is much higher than 20% in case we provide a wrong input, not only a random input.

5.2 Time dependency between scalars and driver's chest acceleration

In this experiment, we investigate which part of the predicted time series is affected by the scalars. For this, we calculate the permutation importance for each scalar without averaging over the entire time series. For each scalar *and for each time step* we calculate the additional MSE error that would occur if we provide a wrong input scalar.

Figure 7 shows the additional error per time step in case of a wrong input scalar. Curves are smoothed by averaging over 10 successive time steps (1 ms). Let's consider the driver::Seatbelt_Used feature. Given the distribution of the error, we notice that this binary feature is affecting the beginning of the output time series. Technically, the seat belt connects the dummy to the car, so that the dummy starts decelerating as soon as possible. We conclude that the model learned perfectly where to inject this scalar feature in the output time series.

In a global view of Fig. 7, we can order the features given the time interval at which they affect the driver's chest acceleration time series:

– driver::Seatbelt_Used (25–50 ms)

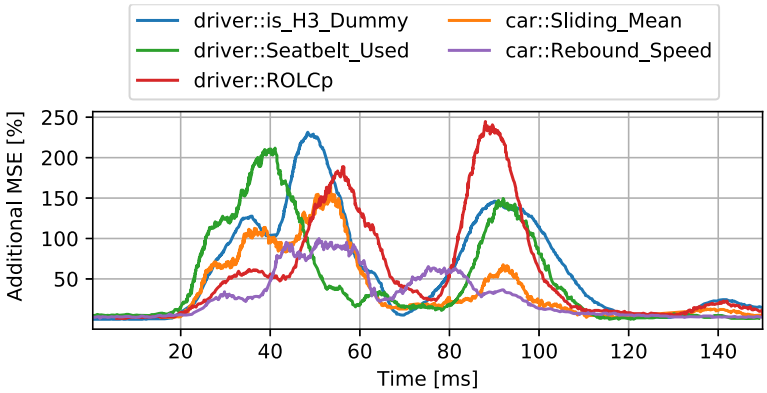


Fig. 7 Feature importance applied on each output timestep separately

– driver::is_H3_Dummy (45–55 ms)
– car::Sliding_Mean (50–55 ms)
– driver::ROLCp (50–65 ms and 85–95 ms)

Note that certain remaining input features did not show any particular pattern: The maximum additional error they generated did not exceed 100% i.e., it did not double the MSE. Thus, we discarded these distributions from the plot to keep only the 5 most impacting features.

5.3 Fully connected layers versus causal padding

In this subsection, we analyze the patterns learned by the output module, which is composed of only one fully connected layer without an activation function. We aim to show that the dense layer learned to rely on past events to predict the occupant's chest acceleration. The equation of the dense layer is $\hat{y} = Wx + b$. The bias (b) and the kernel weights (W) play distinct roles. The bias learns the mean predicted occupant chest acceleration with respect to the input. While the $W \cdot x$ term learns the deviation from it. The x vector represents 22,302 hidden features extracted from the input time series and the scalars. The W matrix is of shape $22{,}302 \times 1500$. For each of the 1500 outputs, 22,302 weights are affecting its value.

Figure 8 is a heatmap representing the amplitude of the kernel weights (W) shared between each input–output pair. For example, the output value at $t = 80$ ms is the scalar product between the extracted features x and the weights represented in the horizontal line at $t = 80$ ms.

When weight amplitudes are close to 0, the output is almost not affected by x, as at the beginning (0−20 ms) and at the end (110−150 ms) of the crash. Whereas, the range 20−110 ms is the output segment, which is mainly affected by the input. In general, a higher weight amplitude is associated with a higher correlation. In the following, we look at four regions of the heatmap in more detail: *past* measures (Region 1), *future* measures (Region 2), and *recent past* measures (Region 3) of the car acceleration and their correlation with the output time series, as well as an interesting artifact (Region 4).

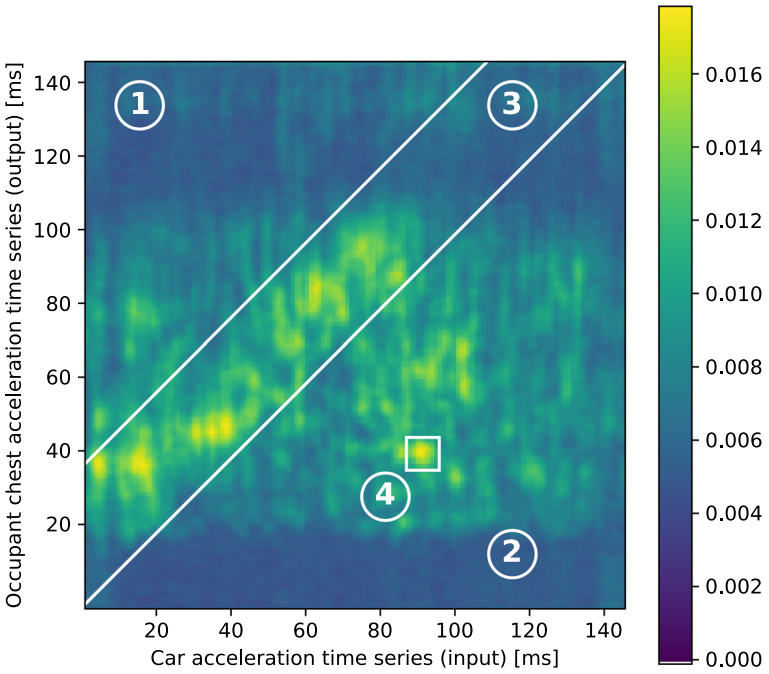


Fig. 8 Impact of the input time series on the occupant chest acceleration

By comparing Region 1 (the upper triangle) to Region 2 (the lower triangle) we notice that the weights of Region 1 have a higher impact on the prediction. Thus, the model is mainly using the past of the car's acceleration time series to predict the dummy's behavior. Moreover, the dense layer learns not only causality but also the number of past steps that must be taken into account. To predict the chest acceleration at a certain time frame t, the model uses the input within the range $(t - 40)$ ms and t. Thus, the last 40 ms is the most valuable segment to predict the actual occupant chest acceleration (Region 3).

Physical models are based only on causality. Whereas data science models exploit any type of correlation, i.e., both causality and observation of the effect. To better understand this, let us consider Region 4, which indicates a high correlation between the dummy's acceleration at 40 ms and the car's acceleration at $t = 90$ ms. Overall, the model is relying on what happened to the car in the future to predict what happened to the occupant a few milliseconds before. After modeling the setup with a spring and masses system, it can be shown that the dummies inside the car (which represent around 10% of the total weight of the car) indeed affect the car deceleration time series. At around 40 ms, the dummy connects to the airbag for the first time and starts transferring its kinetic energy to the airbag that acts as a spring. The kinetic energy becomes potential energy. A few milliseconds later, the energy is transferred gradually from the airbag to the car in the form of kinetic energy, which is observable by analyzing the acceleration of the car. Around 90 ms, the dummy is moving backward.

Mainly, we showed how the model learned by itself how the car is affecting the dummy's chest acceleration and also how the dummy is affecting, in turn, the car acceleration time series. Thus, the model is able to infer the dummy acceleration through the causality effect (car affects the dummy) and observations of the effect of the dummy on the car's acceleration time series.

6 Broader impact

CrashNet is a proof of concept for predicting chest accelerations with neural networks that are able to turn scalars into a time series by learning when to inject them. This architecture can also be applied to predict other time series (knee forces, head acceleration, etc.) and can potentially be applied to different time series prediction tasks. CrashNet is a new tool for automotive safety engineers. It is faster than real crashes and even cheaper than computer simulations.

Moreover, neural networks are based on statistics. They represent a novel approach and we expect it to allow automotive safety engineers to tackle unanswered questions. In the long term, we expect that CrashNet would allow car manufacturers to gradually replace expensive, destructive crash testing. In the short term, we expect that CrashNet will render slow, expensive computer simulations partially obsolete. As a first step, we could perform one real crash to obtain the car pulse and predict the result for the other dummy gender using CrashNet. This would already decrease the costs for car safety development. Nevertheless, a wrong isolated prediction might corrupt the entire car safety development process. Hence, the precision of the results for a specific crash scenario is more important than covering all crash scenarios.

Finally, CrashNet was trained using the dataset of one particular car manufacturer. Nevertheless, we expect that each car manufacturer will fine-tune the model using their own datasets and benefit from accurate CrashNet predictions.

7 Conclusion

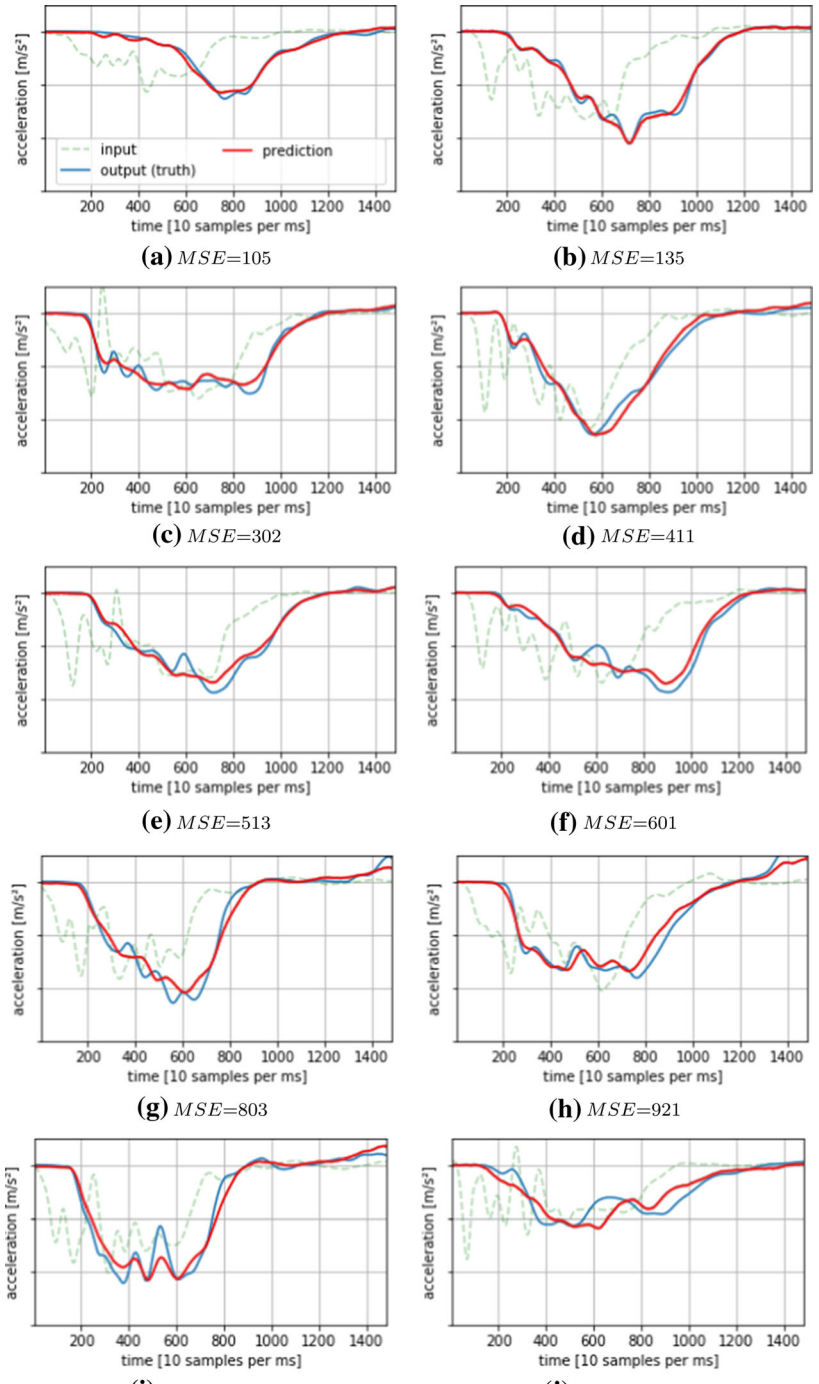
We presented CrashNet, a deep neural network architecture to predict the driver's chest acceleration time series based on scalar input features and the car's acceleration time series. CrashNet is composed of three parts. The encoder processes the car's acceleration time series and the scalar features separately. The decoder merges both and learns how to turn scalars into a time series. The output layer learns the correlation between the high-level features, and the occupant chest acceleration time series. Visualizing the learned weights of the network, we demonstrated that the model learned causal padding by itself resulting in the car's acceleration recorded between $t - 40$ ms and t having the highest influence on the dummy's behavior at time t. Other significant patterns have been observed concerning the effect of the scalar features. Using this model, mechanical engineers could easily vary any feature (e.g., the car mass) and predict its effect on the chest acceleration without any additional cost. We hope that soon automotive safety engineers will benefit from this model to improve the car prototypes especially at an earlier stage of the development process when fewer details

about the car model are needed. CrashNet can assist automotive safety engineers in making better decisions and reducing the number of trial and error iterations thanks to its near-instant predictions. Fewer iterations would finally lead to reducing some of the destructive crash tests or FEM simulations.

Funding Open Access funding enabled and organized by Projekt DEAL.

Annex: Example predictions

Below is a group of predictions. Note that the data is confidential. Thus, the input scalars and the y-axis of the time series remain undisclosed (Fig. 9).



(a) MSE=105 **(b)** MSE=135

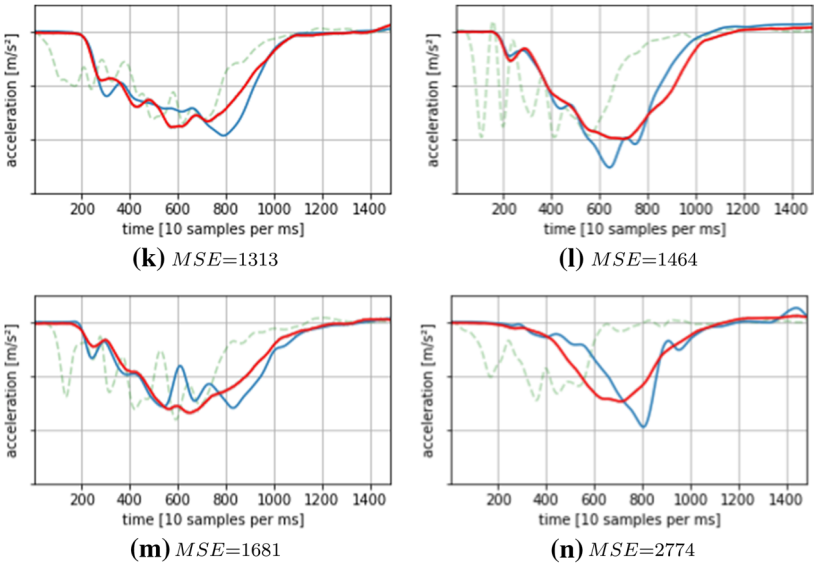
(c) MSE=302 **(d)** MSE=411

(e) MSE=513 **(f)** MSE=601

(g) MSE=803 **(h)** MSE=921

(i) MSE=1110 **(j)** MSE=1234

Fig. 9 Example of predictions from the test set using the CrashNet model



(k) MSE=1313 **(l)** MSE=1464

(m) MSE=1681 **(n)** MSE=2774

Fig. 9 continued

References

Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271

Bastien C, Blundell M, Neal-Sturgess C (2017) A study into the kinematic response for unbelted human occupants during emergency braking. Int J Crashworthiness 22:689–703

Bishop CM et al (1995) Neural networks for pattern recognition. Oxford University Press, Oxford

Bohlien J (2016) Stochastic crash simulations to analyze the influence of joint and assemble scattering on the deformation behavior of vehicle structures under crash. Master's thesis, Universität Stuttgart

Böttcher CS, Frik S, Gosolits B (2005) 20 years of crash simulation at opel-experiences for future challenges

Breiman L (2001) Statistics department. University of California, Berkeley, p 94720

Chen C, Zhang G, Qian Z, Tarefder R, Tian Z (2016) Investigating driver injury severity patterns in rollover crashes using support vector machine models. Accid Anal Prev 90:128–139

Chollet F (2017b) Xception: Deep learning with depthwise separable convolutions. In: Proceedings of the conference on computer vision and pattern recognition. IEEE, pp 1251–1258

Chollet F (2017a) Deep learning with python. Manning Publications Co., New York

Dumoulin V, Visin F (2016) A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285

Fisher A, Rudin C, Dominici F (2019) All models are wrong, but many are useful: learning a variable's importance by studying an entire class of prediction models simultaneously. J Mach Learn Res 20(177):3–81

Fukuoka R, Suzuki H, Kitajima T, Kuwahara A, Yasuno T (2018) Wind speed prediction model using LSTM and 1D-CNN. J Signal Process 22(4):207–210

Grenke BD (2002) Digital filtering for j211 requirements using a fast Fourier transform based filter. SAE Trans 111:359–401

Grunert D, Fehr J (2016) Identification of nonlinear behavior with clustering techniques in car crash simulations for better model reduction. Adv Model Simul Eng Sci 3:1–19

Hilmann J, Hänschke IA (2009) On the development of a process chain for structural optimization in vehicle passive safety. Technical University, Berlin

Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR (2012) Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580

Hollowell WT, Gabler HC, Stucki SL, Summers S, Hackney JR (1999) Updated review of potential test procedures for FMVSS no. 208. NHTSA Docket, pp 6407-6

Ince T, Kiranyaz S, Eren L, Askar M, Gabbouj M (2016) Real-time motor fault detection by 1D convolutional neural networks. Trans Ind Electron 63(11):7067–7075

Iwamoto M, Nakahira Y, Kimpara H, Sugiyama T, Min K (2012) Development of a human body finite element model with multiple muscles and their controller for estimating occupant motions and impact responses in frontal crash situations. Stapp Car Crash J 56:231–268

Kang Z (2005) Robust design optimization of structures under uncertainties. PhD thesis, Universität Stuttgart

Krizhevsky A, Sutskever I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105

Melvin JW (1995) Injury assessment reference values for the crabi 6-month infant dummy in a rear-facing infant restraint with airbag deployment. SAE Trans 104:1553–1564

Park CK, Kan C (2010) Objective evaluation method of vehicle crash pulse severity in frontal new car assessment program (NCAP) tests. Center for Collision Safety and Analysis, George Mason University, pp 15-0055

Rabus M (2019) Prognose von Insassenbelastungen mit Strukturkennwerten. 10 Freiberger Crashworkshop

Ripley BD (2007) Pattern recognition and neural networks. Cambridge University Press, Cambridge

Sivaraman S, Trivedi MM (2009) Active learning based robust monocular vehicle detection for on-road safety systems. In: Intelligent vehicles symposium. IEEE, pp 399–404

Spethmann P, Herstatt C, Thomke SH (2009) Crash simulation evolution and its impact on R&D in the automotive applications. Int J Product Dev 8(3):291–305

Sun Z, Bebis G, Miller R (2006) Monocular precrash vehicle detection: features and classifiers. Trans Image Process 15(7):2019–2034

Szczurek P, Xu B, Wolfson O, Lin J (2012) A platform for the development and evaluation of passive safety applications. In: Intelligent vehicles symposium. IEEE, pp 808–813

Trivedi MM, Gandhi T, McCall J (2007) Looking-in and looking-out of a vehicle: computer-vision-based enhanced vehicle safety. Trans Intell Transport Syst 8(1):108–120

Ullah I, Hussain M, Aboalsamh H et al (2018) An automated system for epilepsy detection using EEG brain signals based on deep learning approach. Expert Syst Appl 107:61–71

Untaroiu C, Adam T (2012) Occupant classification for an adaptive restraint system: the methodology and benefits in terms of injury reduction. IRCOBI Conference Proceedings–International Research Council on the Biomechanics of Injury, pp 205–216

Van Den Oord A, Kalchbrenner N, Kavukcuoglu K (2016) Pixel recurrent neural networks. In: Proceedings of the 33rd international conference on international conference on machine learning, vol 48, pp 1747–1756

van den Oord A, Dieleman S, Zen H, Simonyan K, Vinyals O, Graves A, Kalchbrenner N, Senior A, Kavukcuoglu K (2016) WaveNet: A generative model for raw audio. In: 9th ISCA speech synthesis workshop, p 125

Vangi D, Begani F, Gulino MS, Spitzhüttl F (2018) A vehicle model for crash stage simulation. IFAC-PapersOnLine 51(2):837–842

Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems, pp 5998–6008

Wan R, Mei S, Wang J, Liu M, Yang F (2019) Multivariate temporal convolutional network: a deep neural networks approach for multivariate time series forecasting. Electronics 8(8):876

Will J, Baldauf H, Bucher C (2006) Robustheitsbewertungen bei der Virtuellen Auslegung Passiver Sicherheitssystem und Beim Strukturcrash. Proceedings Weimarer Optimierungs-und Stochastiktage 3

Wojna Z, Ferrari V, Guadarrama S, Silberman N, Chen LC, Fathi A, Uijlings J (2019) The devil is in the decoder: classification, regression and GANS. Int J Comput Vis 127(11–12):1694–1706

Zhao Z, Jin X, Cao Y, Wang J (2010) Data mining application on crash simulation data of occupant restraint system. Expert Syst Appl 37(8):5788–5794