1. Machine learning models

The Logistic Regression (LR) is a classification model that predicts the outcomes by forming a linear combination of the input features followed by a logistic activation function. Because it is a linear model, it requires any non-linear relationships between the input and outcome to be captured by the features themselves.

The multilayer perceptron (MLP) consists of multiple layers of interconnected nodes, called neurons, organized into an input layer, one or more hidden layers and an output layer. Neurons within each layer apply weighted sums and activation functions to their inputs. Through a process called backpropagation, the MLP adjusts its weights iteratively during training to minimize prediction errors. A sufficiently large MLP can in theory model any relationship between the inputs and the outcome.

A convolutional neural network (CNN) is tailored for processing grid-like data such as images, time series data, and more. CNNs incorporate unique layers like convolutional and pooling layers to capture hierarchical features in the data automatically. Convolutional layers employ small filters to scan the input, enabling the network to detect local patterns like edges and textures. Pooling layers reduce the spatial dimensions of the data while preserving essential information. This design reduces the number of parameters compared to fully connected MLPs, enabling CNNs to learn and generalize from large and complex datasets efficiently.

The residual neural network (RN) architecture is an evolution of the original CNN model that addresses the challenges of training very deep neural networks [1]. Residual networks introduce the concept of residual blocks, which consist of shortcut connections that bypass one or more layers. These shortcuts enable the network to learn residual functions, making training easier and preventing the vanishing gradient problem that may occur in deep networks. In contrast to the traditional CNN where each layer learns to transform the input directly, RNs residual blocks learn to adjust the difference between the input and the desired output. This "skip connection" allows for smoother optimization and enables the construction of much deeper architectures while maintaining or improving performance.

For an introduction to deep learning, we recommend the excellent book Deep Learning, by Goodfellow et al. [2].

2. Hyper-parameter optimization

The hyper-parameters for the MLP, CNN and RN models were determined through random search, in which a model is repeatedly trained with randomly sampled parameters and evaluated on the validation set. This process was repeated 400 times for all MLP and CNN models, and 200 times for the RN models. The space of possible parameters that were sampled from is listed in Table 1 for the MLP models, Table 2 for the CNN models and Table 3 for the

RN models. The architecture and parameters of the models that had the best AUC on the validation set are illustrated in Section 3.

Feature extractor	
Number of dense layers	[1, 2]
Neurons	[25, 50, 100, 200]
Dropout rate	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
Batch normalization	[true, false]
Activity L2 regularization weight	[1e-2, 1e-3, 1e-4, 1e-5, 0]
Kernel L2 regularization weight	[1e-1, 1e-2, 1e-3, 1e-4, 0]
Bias L2 regularization weight	[1e-1, 1e-2, 1e-3, 1e-4, 0]
Combiner	
Number of dense layers	1
Neurons	[6, 10, 20]
Dropout rate	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
Batch normalization	[true, false]
Activity L2 regularization weight	[1e-2, 1e-3, 1e-4, 1e-5, 0]
Kernel L2 regularization weight	[1e-1, 1e-2, 1e-3, 1e-4, 0]
Bias L2 regularization weight	[1e-1, 1e-2, 1e-3, 1e-4, 0]
Classifier	
Number of dense layers	1
Neurons	10
Dropout rate	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
Batch normalization	[true, false]
Activity L2 regularization weight	[1e-2, 1e-3, 1e-4, 1e-5, 0]
Kernel L2 regularization weight	[1e-1, 1e-2, 1e-3, 1e-4, 0]
Bias L2 regularization weight	[1e-1, 1e-2, 1e-3, 1e-4, 0]
Additional settings	
Optimizer	Adam
Learning-rate	[3e-3, 1e-3, 3e-4, 1e-4, 3e-5, 1e-5]
Epochs	100
Batch-size	64

Table 1: MLP hyper-parameter search space. The dense layer in the "Classifier" part was only included for models using additional clinical variables. All parameters were sampled uniformly at random from the options shown in the brackets.

Feature extractor	
Number of conv layers	[2, 3, 4]
Pool-size	[[7–30], [4–10], [3–6]]
Filters, first	[8–64]
Filters, last	[8–64]
Kernel size, first	[5–65]
Kernel size, last	[5–65]
Batch normalization	[true, false]
Kernel L2 regularization weight	[0.1, 0.01, 0.001, 0]
Convolutional dropout rate	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
Include final dense layer	[true, false]
Final dense layer neurons	[10, 50, 100]
Final dense layer dropout rate	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
Combiner	
Number of dense layers	[0, 1]
Neurons	[10, 20]
Dropout rate	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
Classifier	
Number of dense layers	1
Neurons	[10, 20, 50, 100]
Dropout rate	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
Additional settings	
Optimizer	Adam
Learning-rate	[3e-3, 1e-3, 3e-4, 1e-4, 3e-5, 1e-5]
Epochs	100
Batch-size	64

Table 2: CNN hyper-parameter search space. The “Feature extractor” part begins with 2, 3 or 4 convolutional blocks, and the pool-size in each case was sampled from the range [7–30], [4–10] or [3–6] respectively. The kernel size and number of filters for each convolutional block was linearly interpolated between the first and last values indicated in the table. Each convolutional block included a potential batch-normalization, followed by max-pooling and dropout. After the convolutional blocks there was a flatten layer, potentially followed by a dense-layer. The dense layer in the “Combiner” part was only used for the models that included additional clinical variables. All parameters were sampled uniformly at random from the options shown in the brackets.

Feature extractor	
Number of dense layers	[1, 2]
Neurons	[25, 50, 100, 200]
Dropout rate	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
Batch normalization	[true, false]
Activity L2 regularization weight	[1e-2, 1e-3, 1e-4, 1e-5, 0]
Kernel L2 regularization weight	[1e-1, 1e-2, 1e-3, 1e-4, 0]
Bias L2 regularization weight	[1e-1, 1e-2, 1e-3, 1e-4, 0]
Combiner	
Number of dense layers	1
Neurons	[6, 10, 20]
Dropout rate	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
Batch normalization	[true, false]
Activity L2 regularization weight	[1e-2, 1e-3, 1e-4, 1e-5, 0]
Kernel L2 regularization weight	[1e-1, 1e-2, 1e-3, 1e-4, 0]
Bias L2 regularization weight	[1e-1, 1e-2, 1e-3, 1e-4, 0]
Classifier	
Number of dense layers	1
Neurons	10
Dropout rate	[0.0, 0.1, 0.2, 0.3, 0.4, 0.5]
Batch normalization	[true, false]
Activity L2 regularization weight	[1e-2, 1e-3, 1e-4, 1e-5, 0]
Kernel L2 regularization weight	[1e-1, 1e-2, 1e-3, 1e-4, 0]
Bias L2 regularization weight	[1e-1, 1e-2, 1e-3, 1e-4, 0]
Additional settings	
Optimizer	Adam
Learning-rate	[3e-3, 1e-3, 3e-4, 1e-4, 3e-5, 1e-5]
Epochs	50
Batch-size	32

Table 3: RN hyper-parameter search space. The dense layer in the "Classifier" part was only included for models using additional clinical variables. Two learning-rates were sampled for each model, and the higher rate was used for the first n epochs (where n was a random number between 5 and 25), after which the lower learning rate was used. All parameters were sampled uniformly at random.

3. Final model parameters

3.1. Logistic regression models

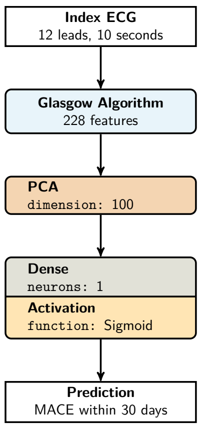

Figure 1: Parameters and structure of the logistic regression model using only the index ECG as input.

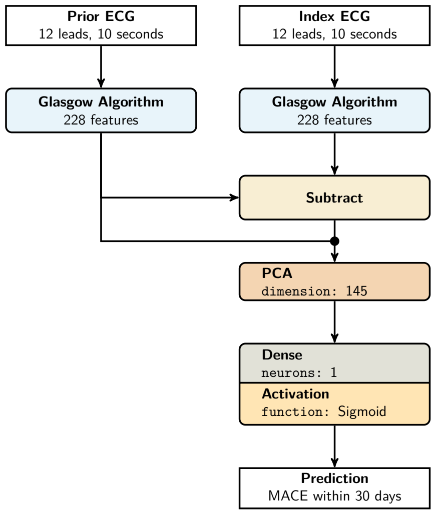

Figure 2: Parameters and structure of the logistic regression model using both the index and prior ECGs as input. Subtraction is vector-valued. The circle denotes concatenation.

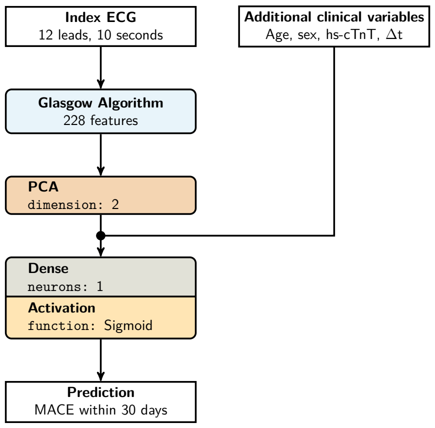

Figure 3: Parameters and structure of the logistic regression model using index ECG and additional clinical variables as input. Δt is the time between the two ECGs. The circle denotes concatenation.

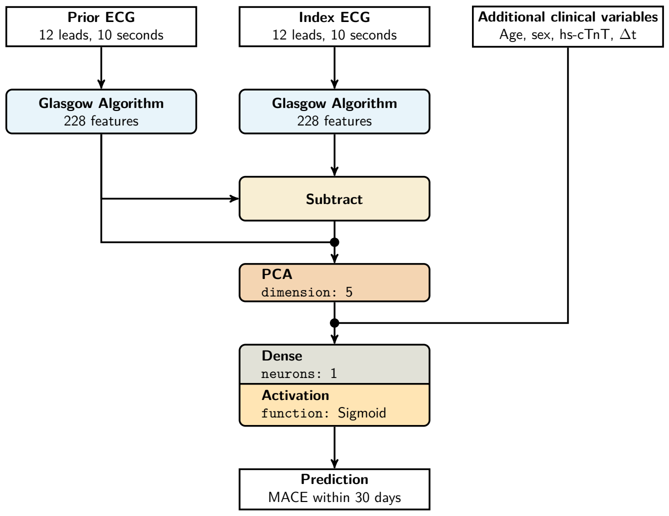

Figure 4: Parameters and structure of the logistic regression model using both index and prior ECGs together with additional clinical variables as input. Δt is the time between the two ECGs. Subtraction is vector-valued. The circle denotes concatenation.

3.2. Multilayer perceptron models

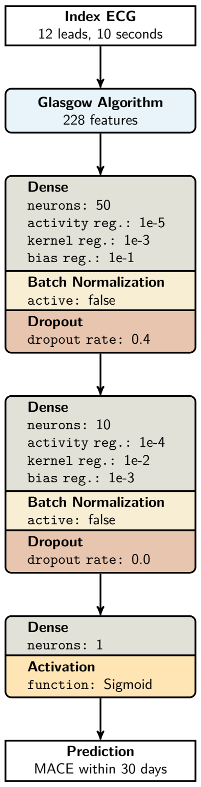

Figure 5: Parameters and structure of the multilayer perceptron model using only the index ECG as input. The learning rate was 0.0001.

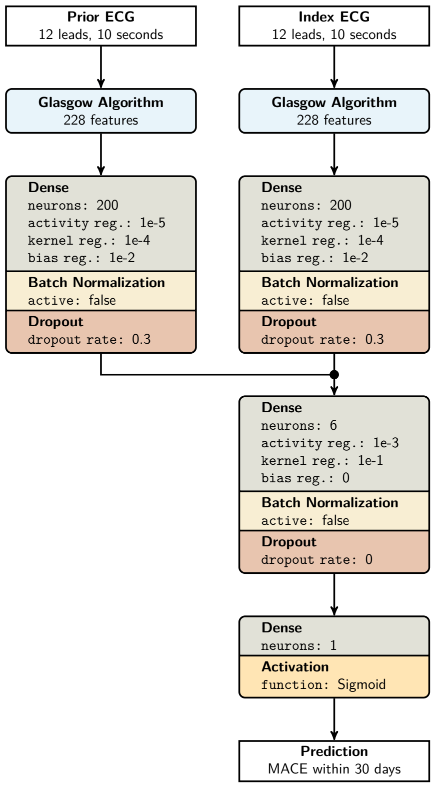

Figure 6: Parameters and structure of the multilayer perceptron model using both the index and prior ECGs as input. The circle denotes concatenation. The learning rate was 0.0001.

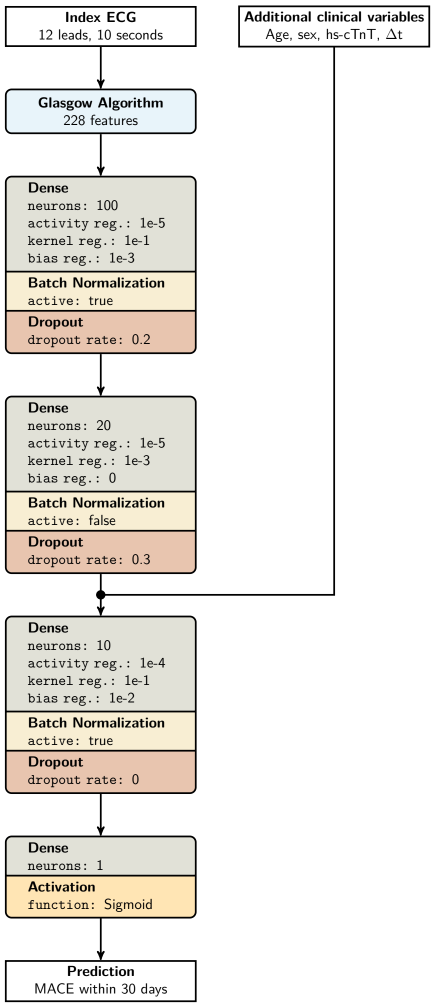

Figure 7: Parameters and structure of the multilayer perceptron model using index ECG and additional clinical variables as input. Δt is the time between the two ECGs. The circle denotes concatenation. The learning rate was 0.0003.

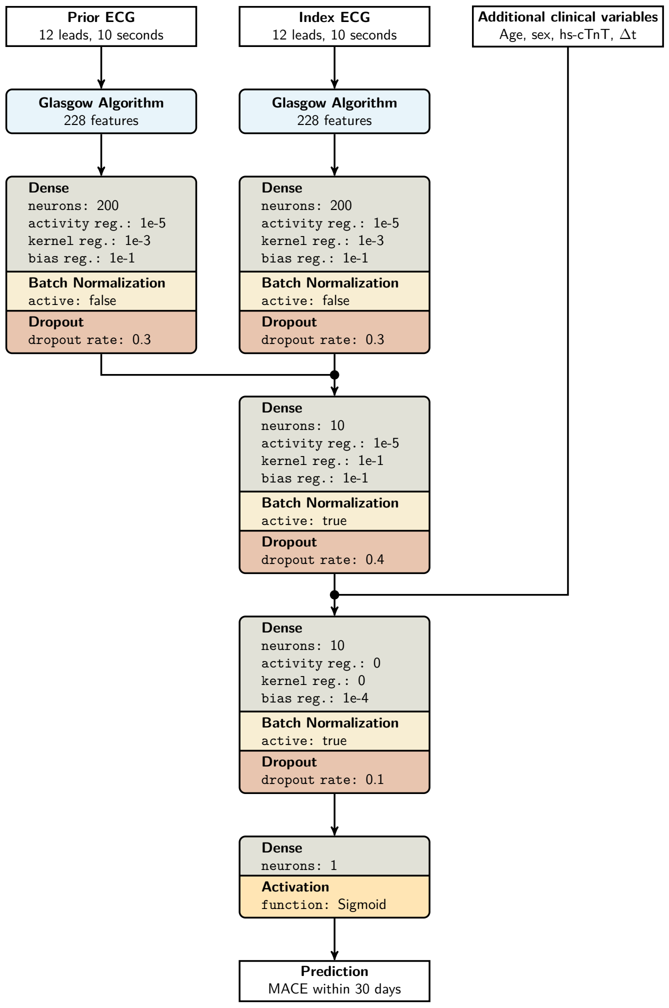

Figure 8: Parameters and structure of the multilayer perceptron model using both index and prior ECGs together with additional clinical variables as input. Δt is the time between the two ECGs. The circle denotes concatenation. The learning rate was 0.001.

3.3. Convolutional neural network models

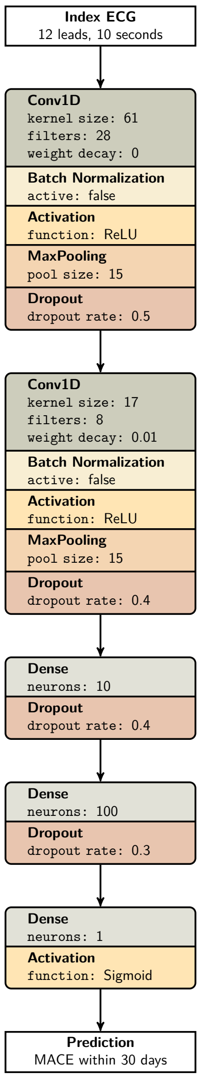

Figure 9: Parameters and structure of the convolutional neural network model using only the index ECG as input. The learning rate was 0.0001.

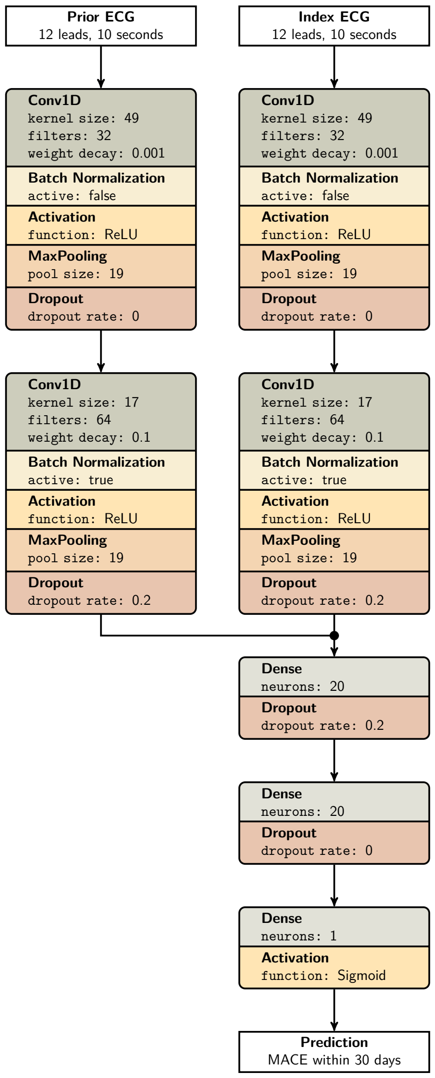

Figure 10: Parameters and structure of the convolutional neural network model using both the index and prior ECGs as input. The circle denotes concatenation. The learning rate was 0.00001.

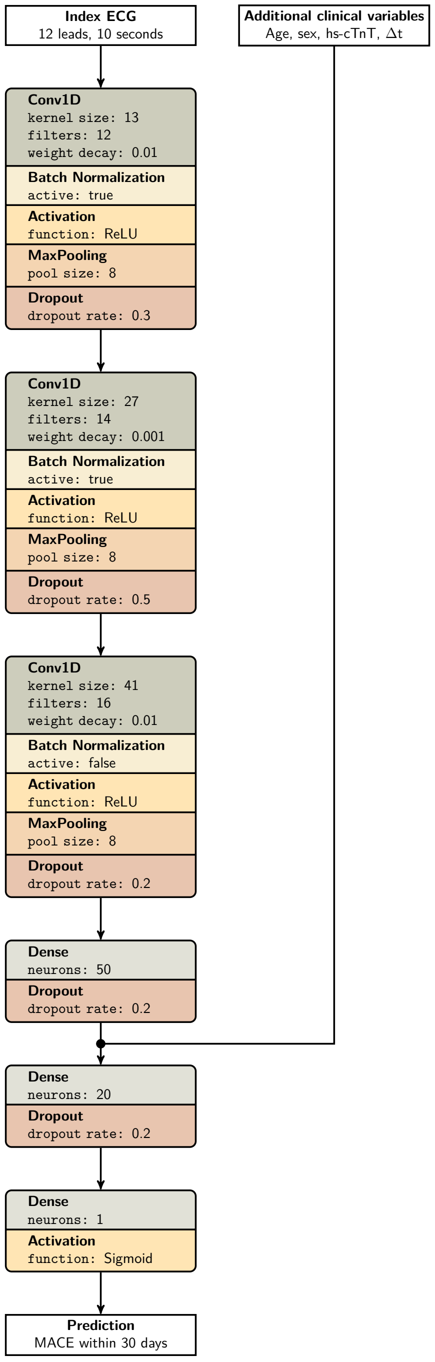

Figure 11: Parameters and structure of the convolutional neural network model using index ECG and additional clinical variables as input. Δt is the time between the two ECGs. The circle denotes concatenation. The learning rate was 0.001.

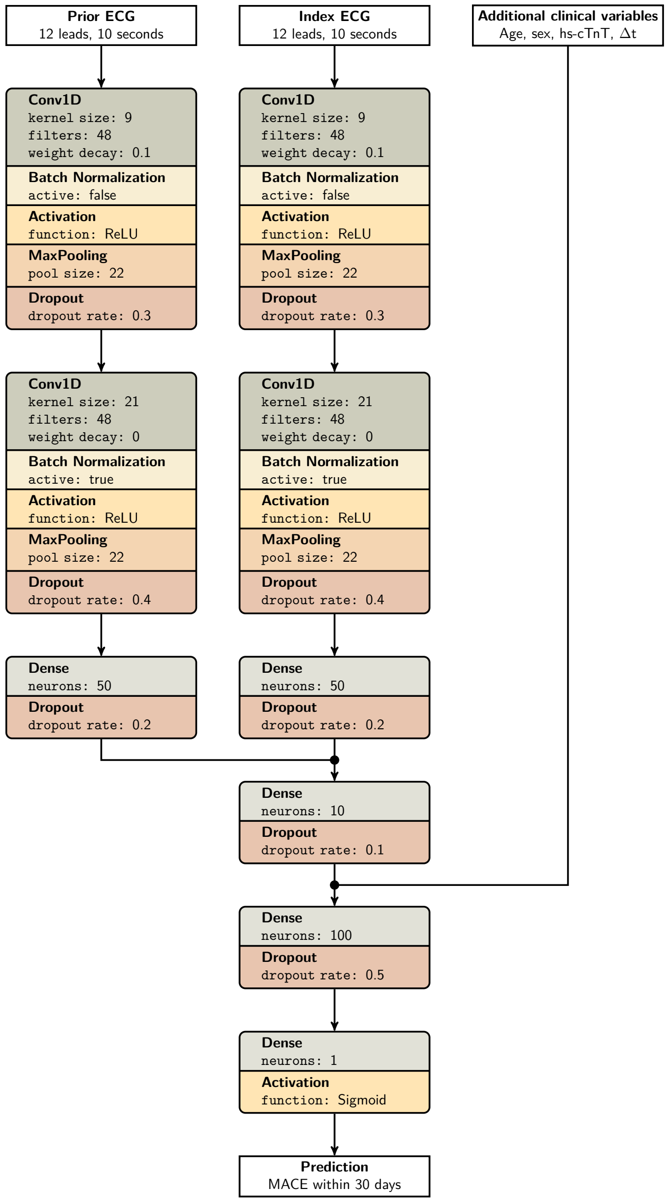

Figure 12: Parameters and structure of the convolutional neural network model using both index and prior ECGs together with additional clinical variables as input. Δt is the time between the two ECGs. The circle denotes concatenation. The learning rate was 0.001.

3.4. Residual neural network models

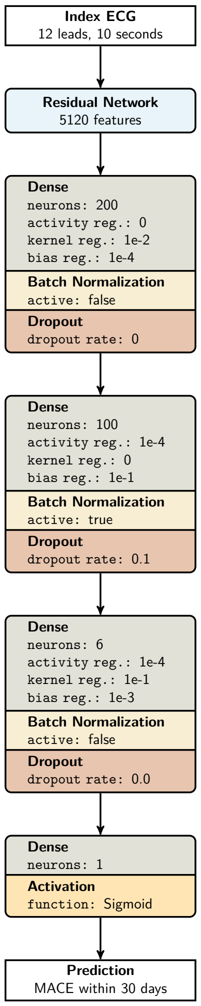

Figure 13: Parameters and structure of the residual network model using only the index ECG as input. The learning rate was 0.0003 for all 50 epochs.

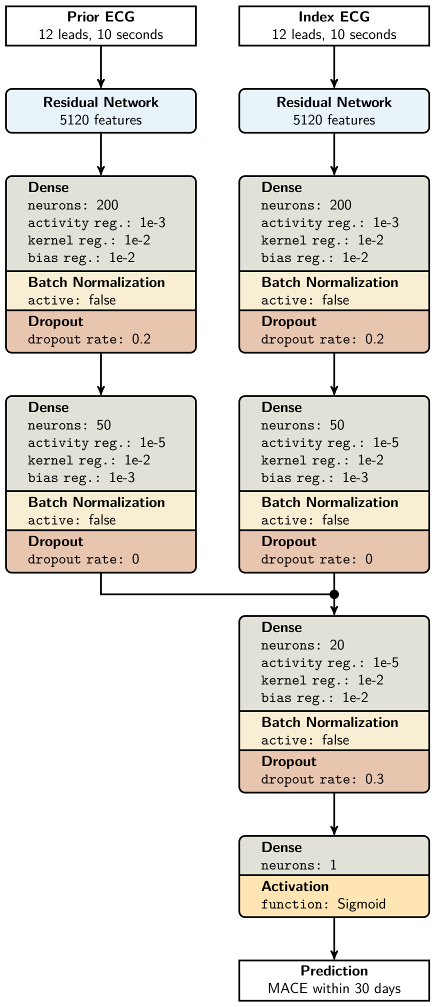

Figure 14: Parameters and structure of the residual network model using both the index and prior ECGs as input. The circle denotes concatenation. The learning rate was 0.001 for the first 18 epochs, and 0.00001 for the final 32 epochs.

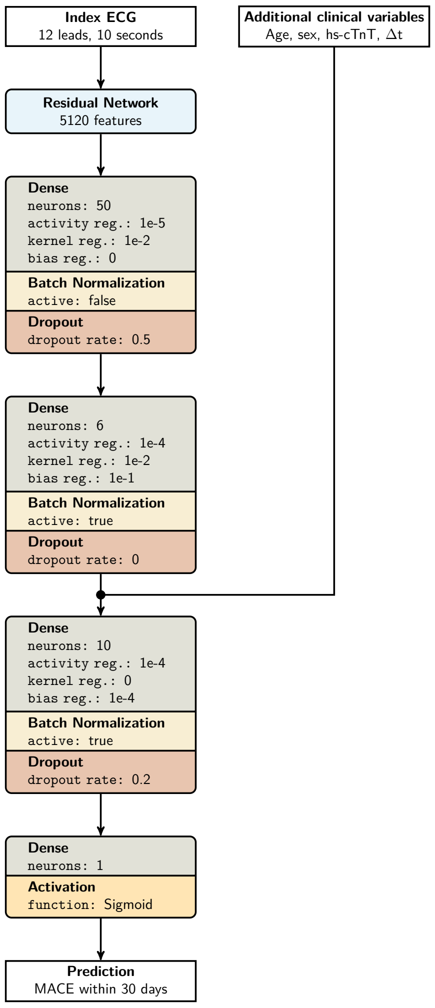

Figure 15: Parameters and structure of the residual network model using index ECG and additional clinical variables as input. Δt is the time between the two ECGs. The circle denotes concatenation. The learning rate was 0.0003 for the first 24 epochs, and 0.00003 for the final 26 epochs.

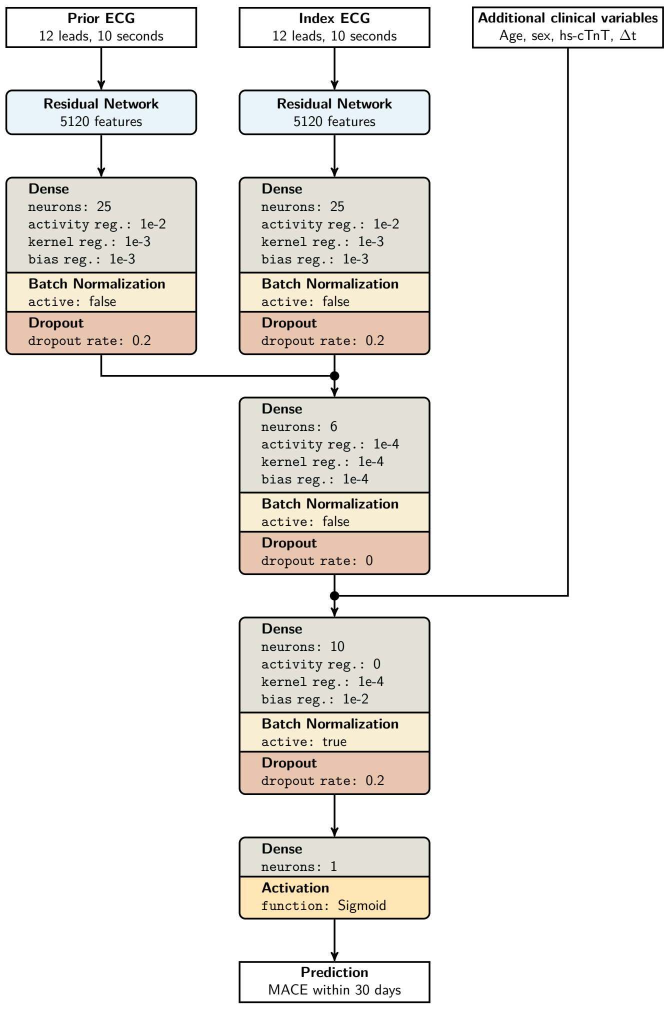

Figure 16: Parameters for and structure of the residual network model using both index and prior ECGs together with additional clinical variables as input. Δt is the time between the two ECGs. The circle denotes concatenation. The learning rate was 0.003 for the first 33 epochs, and 0.00001 for the final 17 epochs.

References

[1] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[2] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016. `http://www.deeplearningbook.org`.