**Building the convolutional neural network model**

To capture the characteristics of the nEMG signal, we designed a 1-dimensional convolutional neural network (CNN). The structure of nEMGNet includes 3 types of convolutional blocks; spatial block-1, which reduces the spatial resolution by half, spatial block-2 which reduces the spatial resolution by quarter, and residual block which solves the vanishing gradient problem which happens as the number of layer gets deeper. (S1 Table)

|  |  |  |
| --- | --- | --- |
| **Spatial reduction block-1**  **(n,k)** | **Spatial reduction block-2**  **(n,k)** | **Residual block**  **(n)** |
| Conv(k)-n,  Stride(1) | Conv(k)-n,  Stride(2) | Conv(5)-n,  Stride(1) |
| BatchNorm | BatchNorm | BatchNorm |
| ReLU | ReLU | ReLU |
| Max-pool(2),  Stride(2) | Max-pool(2),  Stride(2) | Conv(5)-n,  Stride(1) |
|  |  | BatchNorm |

S1 Table. convolutional blocks of nEMGNet. n, the number of channel; k, the number of filter; conv, convolutional layer; BatchNorm, batch normalization; Max-pool, max pooling.

The nEMGNet was tested with 4 versions of nEMGNet-A, nEMGNet-B, nEMGNet-C, and nEMGNet-D with different versions according to the number of residual blocks. (S2 Table)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **nEMGNet-A** | **nEMGNet-B** | **nEMGNet-C** | **nEMGNet-D** |
| Block 1 | SR block-2  (64, 11) | SR block-2  (64, 11) | SR block-2  (64, 11) | SR block-2  (64, 11) |
| Block 2 | SR block-2  (64, 7) | SR block-2  (64, 7) | SR block-2  (64, 7) | SR block-2  (64, 7) |
| Block 3 | SR block-2  (64, 5) | SR block-2  (64, 5) | SR block-2  (64, 5) | SR block-2  (64, 5) |
| Block 4 |  | Residual block  (64) 2 | Residual block  (64) 4 | Residual block  (64) 6 |
| Block 5 | SR block-1  (128, 5) | SR block-1  (128, 5) | SR block-1  (128, 5) | SR block-1  (128, 5) |
| Block 6 |  | Residual block  (128) 2 | Residual block  (128) 4 | Residual block  (128) 6 |
| Block 7 | SR block-1  (256, 5) | SR block-1  (256, 5) | SR block-1  (256, 5) | SR block-1  (256, 5) |
| Block 8 |  | Residual block  (256) 2 | Residual block  (256) 4 | Residual block  (256) 6 |
| Block 9 | SR block-1  (512, 5) | SR block-1  (512, 5) | SR block-1  (512, 5) | SR block-1  (512, 5) |
| Block 10 |  | Residual block  (512) 2 | Residual block  (512) 4 | Residual block  (512) 6 |
| Block 11 | SR block-1  (1024, 5) | SR block-1  (1024, 5) | SR block-1  (1024, 5) | SR block-1  (1024, 5) |
| Block 12 |  | Residual block  (1024) 2 | Residual block  (1024) 4 | Residual block  (1024) 6 |
|  | FC-512 | FC-512 | FC-512 | FC-512 |
|  | FC-256 | FC-256 | FC-256 | FC-256 |
|  | FC-64 | FC-64 | FC-64 | FC-64 |
|  | FC-16 | FC-16 | FC-16 | FC-16 |
|  | FC-10 | FC-10 | FC-10 | FC-10 |
|  | Softmax | Softmax | Softmax | Softmax |

S2 Table. configuration of 4 different versions of nEMGNet. SR block, spatial resolution block; FC, fully connected layer.

A rectified linear unit (ReLU) is applied to the fully connected layer after the convolutional layer. (Figure 1)



Figure 1. nEMGNet structure. Processing composed of convolutional neural network, batch normalization, rectified linear unit (ReLU), max pooling. Light gray box; spatial block-1 and spatial block-2, Dark gray box and bold curved arrow; residual block. 유재성연구원 그림 바꿔서 주면 변경

The initial values of nEMGNet hyperparameter were empirically determined based on values that have been widely used. The learning rate, batch size, and epoch were set to 10-3, 32, and 100. Adam optimizer was used for optimizer, and inversely proportional values were used for the class weight to the number of signal segments for preventing erroneous prediction.

After the nEMGNet was trained, the signal segments which nEMGNet best classifies as neuropathy, myopathy, and normal were derived using feature visualization technique. The signal segments were generated from optimizing noise using Adam optimizer for gradient descent. Random jitter regularization was applied to the optimization process to generate realistic signals. We used feature visualization to identify the characteristics of each classified waveform learned through nEMGNet.

**Divide and Vote (DiVote) algorithm**

The number of muscles tested with nEMG is different for each subject, and among the tested muscles, abnormal and normal nEMG can coexist. To overcome these limitations, we applied a method called the DiVote (Divide and Vote) algorithm. DiVote divided each nEMG signal into segments of homogeneous length and converted it into 3 signal segment prediction scores through a feature extractor. The muscle signal prediction score was calculated by aggregating the signal segment prediction score and aggregated to derive the subject prediction score through soft voting. When deriving the subject prediction score, two different method were tried. The first method is to calculate and aggregate prediction scores by classifying them by neuropathy, myopathy, and normal without information on the location of the muscle. The second method is to classify whether it is proximal or distal according to the location of the muscle, and calculate and aggregate prediction scores of each neuropathy, myopathy, and normal.

There two major methods of aggregated probabilities corresponding to neuropathy, myopathy, and normal of the segment from nEMG signal data for each muscle, hard voting and soft voting. The former is to select one item with the highest probability for each segment, count the number of selected items and finally select the one that occupies the largest number among them. On the other hand, the latter is to select items with highest probabilities after averaging each probability from each segments. In the case of hard voting, where only values with high probability are selected for each segment, data loss might occur, so soft voting was used to reduce data loss. The most probable diagnosis obtained through soft voting in the segment is the muscle prediction score, and the subject prediction score is the soft voting of all muscles by subject after soft voting in the segment.

The classifier to distinguish neuropathy, myopathy, and normal was obtained in two ways and the classified results were compared. First, it was determined through logistic regression analysis with the subject features of each group as the independent variables and the diagnosis as the dependent variable; Second, it was determined by additionally considering muscle locations as an independent variables.

**Development of physicians’ labeling platform: EMG Labeler**



S1 Figure. An example of nEMG waveform data shown on EMG Labeler

 We developed a web-based labeling platform to acquire the clinicians’ reading decisions and dubbed it as ‘EMG Labeler’ (Figure S1). Thus, it was available to collect the decisions regardless of time, place, device and operating systems. For the sake of compliance towards Korean Guidelines for Utilization of Healthcare Data (PIPC & MOHW\*) and institutional guidelines, we de-identified the patient identification number into a random number. We stored the waveform data in the Redis database and labels the PostgreSQL database in order to get advantage of key-value storing and relational database engine. During the label acquisition, we used an ordinary server composed of 20 threads and 64 GiB RAM; the operating system was Linux distro: CentOS v7. The web engine is based on Node.js v16, which utilizes Google V8 engine, is generated from C++ codes and makes web pages speedy by compiling the JavaScript source code natively. Moreover, we adopted the WebGL-based GPU rendering to plot the waveform in a high frequency; pseudo-threading to synchronize the waveform movement with sound and transaction in the labeling button to prevent users from labeling the data while the service is broken.