Diagnostics support of musculoskeletal diseases using artificial neural network

1st Zdenek Novotny
Faculty of Electrical Engineering and Informatics
University of Pardubice
Pardubice, Czech Republic
st46619@student.upce.cz

2nd Jan Mares
Faculty of Electrical Engineering and Informatics
University of Pardubice
Pardubice, Czech Republic
jan.mares@upce.cz

3rd Petr Dolezel
Faculty of Electrical Engineering and Informatics
University of Pardubice
Pardubice, Czech Republic
petr.dolezel@upce.cz

***Abstract*—A vestibular schwannoma is a benign tumor, developing in the inner ear. As it grows, it may affect patient's hearing and body balance. If not treated, it can also lead to death of the patient. Once it becomes a problem, it is surgically removed. During the surgery, there is a high risk that surrounding nerves become harmed (it causes problems with facial movement). This document discusses evaluation of such injury, based on a modern approach of classification using artificial neural network.**

***Index Terms*—biomedicine, diagnostics, Keras, musculoskeletal diseases, Python, Tensorflow, artificial intelligence, artificial neural network, big data.**

I. Introduction

A. Vestibular schwannoma

The cause of a vestibular schwannoma is overproduction of Schwann cells. These are normally wrapping nerves, but when over-produced, they form a tumor. A tumor usually grows slowly, and it is mostly observed, when it is small. While growing, it can affect surrounding nerves and lead to several harmful effects. These include, for example, partial deafness, ringing in the ear, loss of balance, as well as facial paralysis and asymmetry. These complications may seriously affect the patient's life.

If it grows critically, it starts to press on the brain, which can also cause death of the patient. Further impacts and details of such a tumor are discussed in [1].

If the swannoma causes problems, it is usually surgically removed. While removing, it is necessary to cut the vestibular nerve and surrounding nerves are easily harmed, too. Thus, the surgery itself can make symptoms even worse [1]. Surrounding nerves include facial nerves, whose injury leads to facial paralysis, articulation disorder and non-verbal communication problems [1].

After the surgery, a patient participates in physical therapy, which takes several months. The therapy includes evaluation of facial moveability in order to assess the damage. The first evaluation takes place before the surgery, to set the reference value to which post-surgery evaluations can be compared. Post-surgery evaluations are usually performed a week after the surgery, a month later and three months later [2].

These days, the evaluation depends on a doctor, who visually checks the patient, performing a set of facial exercises. As the result is bound to be subjective, there is a need for automation of this process. A computer is able to evaluate the patient objectively and may also declare a unified scale. Furthermore, it could set a base for diagnostics without the need of a physician, for example at home.

There are several approaches that can be taken to make the automation of facial exercise evaluation possible. This document focuses on machine learning, using artificial neural networks to dynamically evaluate face points' positions in time. The points are further described in [2].

B. Artificial neural networks

Artificial neural networks are based on artificial neurons, which are defined in [3]. Each neuron processes its input signals and creates an output signal. By mutual interaction of such neurons, more sophisticated predictions can be achieved. The fundamental advantage, in comparison to expert systems, is its ability to learn, while all it needs are input and output values. It is not necessary to define the inner process of evaluation. On the other hand, its advantage over stochastic algorithms is computation speed. While the computation in stochastic algorithms is focused once the input is defined, artificial neural networks are demanding especially at the learning phase. A prediction for a given input is then usually fast and efficient.

That is the reason why artificial neural networks are applied in many fields, including banking, traffic, electronics, medicine, social engineering, telecommunications, industry, education and many more. It solves data classification, cluster analysis, pattern recognition and prediction [3].

There are many known topologies for classification of time series. Quality of the output is partially dependent on input - its shape, signal noise and potential dependency of its ingredients. Some well-known networks include:

- Feed-forward neural network - one of the mostly used artificial neural networks. It is based on simple neurons, which are layered. All the neurons

The work has been supported by the SGS grant at the Faculty of Electrical Engineering and Informatics, University of Pardubice, Czech Republic. This support is very gratefully acknowledged.

ISBN 978-80-261-0973-0,

in a layer are connected to all the neurons in the next layer. Connections are neither present between neurons in the same layer nor there are any backward connections. Feed-forward neural networks are mainly used for approximations and predictions. Their disadvantage for time series classification is that the coherence of timesteps is lost [4].

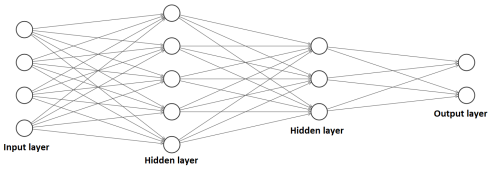

Fig. 1. Feed-forward neural network.

- Convolutional neural network - another widely used approach, mainly in image analysis. Convolution is an important operation in signal theory. Values are aggregated over a sliding window, so minor deviations are suppressed. It is further explained in [5]. Convolution layer is commonly followed by a pooling layer, which aggregates the data even more, reducing noise and shape [5]. Usually a max pooling (resulting in max-values) or average pooling (resulting in average values) is used. Convolution and pooling together can significantly reduce shape of input while preserving important features. The resulting data are then usually analyzed by a feed-forward neural network [4] [5].

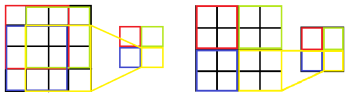

Fig. 2. Convolution and pooling.

- Recurrent neural network - recurrent neural network is equipped with a temporary memory. Its output is based on a current input, as well as historic data. That is the reason why these are commonly used for prediction of dynamic values, that may depend on preceding progress. Even though the recurrent neural network is preferably designed for next-step prediction, there are known experiments of using it as a time series classifier [4], [6].

C. Artificial intelligence in diagnostics

Artificial intelligence has already been used in medicine to solve various problems. For example, it is used for spectral analysis, modelling of mental defects, predictions of disease progress, and above all, for analysis of image data and signal processing.

The University of Chemistry and Technology in Prague, in conjunction with University Hospital Kralovske Vinohrady, developed a system for post-surgery diagnostics of muscoskeletal diseases using image processing. It consists of robotic scanning of patient's walk and static scanning of his or her face [2]. These image data are then analyzed. The experiment, discussed in this document, is based on their data.

II. Problem formulation

The aim of this article is to develop a diagnostic support of post-surgery diagnostics of muscoskeletal insufficiency during physical therapy. As an input to the development, data about patients and their diagnostics were provided by our partners from the University Hospital Kralovske Vinohrady.

A. Image data

Firstly, image data are needed. In this experiment, these data were captured by a Microsoft Kinect sensor, which is able to take a stereoscopic visual record. The sensor has been placed statically, capturing the face of the respective patient. The patient then performed a total of nine facial exercises (for example to raise his or her eyebrows, to smile, to scowl, etc.). The record was consequently turned into a three-dimensional model of the patient's head. From the model, twenty-one points composed the input for classification. As the model is three-dimensional, each point is defined by three Cartesian coordinates and time. Mentioned points are illustrated in the figure below.

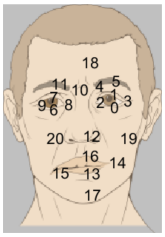

Fig. 3. Captured points.

B. Diagnostic data

Another important data is the original evaluation, as decided by a doctor. Each patient is assigned a mark from 0 to 9, which represents his or her overall evaluation. The higher the mark, the more are patient's nerves damaged.

As the set of patients is small, each patient evaluation was processed independently. As such, connections between examinations of the same patient (which are not desired anyway) were lost, but the data made proper input for an artificial neural net.

For the learning phase, it is necessary to prepare a lot of samples, covering the whole range of evaluations. However, only 89 samples were provided. In addition, almost half of them were evaluated with mark 1. In contrast, there were no evaluations with mark of 0, 7, 8 or 9 at all.

III. Proposed solution

According to the previous authors' experience, Python programming language with Tensorflow-Keras framework was selected to develop the solution.

For the implementation, a proper data format was crucial. The final format used is as such: Incomplete

patients (patients who had not done some of the exercises) were dropped.

As a capturing period of the used sensor should be nearly a constant, the time stamps were dropped too. The time flow was substituted by timesteps. Thus, a single point was represented by a matrix of Cartesian coordinates x, y, z and timesteps.

A single exercise is a matrix of such points and a single patient is a matrix of exercises. As there were nine exercises of twenty-one observed points, this results in a matrix with 567 columns and rows equal to timesteps of the respective exercise.

As the used artificial neural network had fixed size, it was necessary to adjust the size to the longest time series of all exercises.

Also, some transformations were considered and applied. One reason for transformations is the need to comply with Keras input, the other is for optimisation, as there is just a few samples and any improvement can help.

To meet the input, it is necessary to define the input layer of the artificial neural network. That is a part of the network's topology design, where the number of layers, number of neurons and the output layer are defined. There is no ideal topology, one must undertake some experiments to make the decision. However, with enough resources, these experiments can be automatized (e.g. evolutionary algorithm).

A. The input

The data format described above is pretty simple and understandable for a human being. But the Keras framework has different needs. If only one input is present, it must be a TensorFlow tensor, a NumPy array or an array-like structure. In case of multiple inputs, the Keras framework expects a list of these structures. Each item of the list then represents the whole input set (or generated batch) of an input filter.

Thus, it is mandatory to create such a list. First, let us unify the exercises dimensions. The width of such array is given by number of points, multiplied by 3 dimensions in space. The length, on the other hand, differs by the number of captured timesteps. To unify the length, respective exercises have been arranged into an array with length of the longest exercise of them all. Shorter exercises have been filled by zeros from the top.

It was also considered useless to carry the timestamp of each step, as the capture speed was constant and the same for each exercise and therefore the time flow of the respective exercise can be inferred from the number of its timesteps.

Such arrays can then be concatenated to a sole array, which represents all exercises of a respective patient. The issue of data format is solved, but as the data sample is small, there were also some optimisation processes applied to increase the change of success.

Firstly, as the data are captured by a camera, reference point positions are bound to be affected by the respective patient's stance. This means that the actual position of a point consists of an initial value and an increment, while the motoric functions are denoted by the increment only. To eliminate the anomaly in the initial values, the initial value of a point in each dimension has been subtracted from every corresponding value of timesteps. Like that, the resulting array consists just of increments.

Another optimisation targets the set division. Later, during testing, it is necessary to train the net. The quality of training also needs to be evaluated, which is the purpose of the validation set. The validation set is usually approximately a quarter of the input data. The rest forms the training set. But the original data were unbalanced and some of the output evaluations were entirely missing. In fact, almost half of patients were evaluated by mark 1. On the other hand, none were evaluated by marks 0, 7, 8, or 9. If divided randomly, there was a large probability that the low-count evaluations would be missing in either set. To ensure that the training set will be enough, it was divided with an accent to contain at least seventy-five percent of each evaluation.

The last optimisation process inheres in change of the sampling rate. As the input data are basically a collection of signals, it suffers from distortion. The distortion can be repressed by lowering the sampling rate. With lower sampling rate, only serious changes are apparent. On the other hand, some important details would be lost, so it is generally hard to determine the right sampling rate. That is why the original data were preserved too, along with down-sampled versions of a half and one fifth. This means three input arrays, for three input filters of the net.

Also, although the Keras framework can train the net using a single array and thus it could use the whole training set, it is better to divide it into batches. As there are more patients expected in future, the training set is going to grow. Training of an artificial neural network can be very storage-demanding and this can lead to an input so large, that it will exhaust the whole memory. To reduce such risk, smaller input batches were used for training. The generator, implemented for creation of such batches, outputs four (optionally shuffled) lists, divided to batches. The first list contains patient data of original sampling rate, the second one contains half-sampled data and the third one fifth-sampled. The last one is the list of corresponding evaluations.

B. The network

In this experiment, a convolutional artificial neural network has been chosen to classify the input. Convolutional neural networks are mainly used in image processing, but they can also be used as time series evaluators. The main advantage of such neural network is its ability to learn strong features, resulting in a chance to distinguish between motoric disfunctions and distortion. Additionally and simultaneously, a feed-forward artificial neural network has been used as another classificator for comparison. Feed-forward neural networks are known as a good predictors, but its topology ignores coherence of respective timesteps.

The differences between both nets are mainly in hidden layers, as the input layer is bound to fit the input data and the output layer must fit the expected result format.

The input layer consists of three input points. Each input point is basically a standalone artificial neural network. Each such network has its own input layer, which has to fit the input data. Therefore, it differs in length, according to sampling rate, as mentioned above. Then, convolution follows. In this experiment, one-dimensional convolution has been chosen, as it is not desired to move the filter in a single timestep. In contrast, it is necessary to slide the filter over timesteps from top to bottom, which is just what the one-dimensional convolution does. Apparently, the width of the filter was set to the same as the input layer's width. There are several parameters to consider in connection with a convolutional layer:

- Filter count - generally, the more filters, the more features can the network learn. It is hard to define the right count and this parameter was a subject of examination.
- Filter length - in this particular experiment, as mantioned earlier, this means the number of timesteps in one convolution step. This parameter was a subject of examination as well.
- Padding - set to "same", which means an overlap of zeros is added to the input data, resulting in the same rows in the output as there were in the input.
- Activation - set to "tanh", hyperbolic tangent. This function is continuous on (-1; 1), so there is no need to normalize the output later.

After the convolution, a pooling layer follows. Namely a one-dimensional global max pooling was chosen to keep just a single, strongest feature of each filter. The result is then processed by a feed-forward layer. This layer has following parameters to be set:

- Neurons count - theoretically, the more neurons in a layer, the better prediction. However, there is a limit, where additional neuron improvement is not worth the cost. The right value was a subject of examination.
- Activation - hyperbolic tangent was used there too.

The feed-forward layer is followed by a dropout layer, which deactivates random neurons while training, and weakening relations between them. This helps to avoid learning "by rote".

The described standalone networks form the main input layer. All the outputs are then concatenated into a single array, using a concatenate layer. Then they are processed by a hidden layer, which is another feed-forward neural network.

The last layer is the output layer, represented by further feed-forward layer. However, the output layer is not of the same format as those previous feed-forward networks. It must fit the expected output, which means discrete marks of $< 0; 9 >$. This can be accomplished by classification. Each evaluation mark is represented by a class and the network must classify the input data to one of them.

The usual approach of classification is the softmax activation function. Simply put, softmax computes the likeness of an input to a class. The likeness is represented by a rational number of (0; 1) and the sum of such likenesses equals 1. As it is mandatory to compute the likeness for each class, the number of neurons equals the number of classes. Meaning 10 of them in this experiment.

It is also necessary to choose a loss function. The loss function reflects how much the actual predicated output missed the expected result. In classification, cross-entropy functions are usually used. A cross-entropy function indicates high loss value especially if a wrong class has been chosen with high confidence. As there is always just one correct class in this experiment, meaning a patient should be evaluated by just one mark, categorical cross-entropy loss function was used, which meets these requirements. Another training output is the accuracy. While the loss value quantifies divergence of expected output, accuracy quantifies the rate of correct classifications. Generally, a low loss value means a better net, while high accuracy denotes that it is trained well. Optimisation of these values is just what this experiment was all about.

As already mentioned, a feed-forward neural network has been used for comparison. It has been implemented similarly. The difference is that instead of convolutional and pooling layers, another fully connected layer has been used.

C. The test

With input data in the desired format and networks ready for training, it was time to move to experiments. These experiments have several goals to accomplish:

- Prove that the artificial neural network is functional.
- Prove that it is possible to diagnose musculoskeletal diseases using artificial neural network.
- Find out if the diagnostic is accomplishable at present.
- Choose the final parameters of the network.

The network is functional, if it predicts output for a given input and if it is capable of training itself with the prepared generator. The diagnostic is possible, if the network's training impacts its classification ability, meaning it can find and learn coherence in the data. This is currently possible, if the training enhances the classification ability so much that the network closely predicates correct evaluations of the validation set. The last goal was to find the final parameters of the network. These are, in particular, the number of filters and neurons the of hidden layer and also the number of hidden layers. The indicator of training quality has been set as minimisation in case of loss function and as maximization in case of accuracy. These experiments were executed with both the convolutional neural network and the feed-forward neural network. Each training consists of 1000

learning epochs. These parameters changed a bit later, according to optimisation of parameters.

IV. Results and discussion

Results of all experiments were examined and written down into several tables. In initial experiments, the number of neurons varied.

TABLE I
1 layer, 10 filters, 10 neurons, filters of length 8, 16 and 24

Net	Set	Metric	Expt. 1	Expt. 2	Expt. 3
FFNN	Training	Loss	0.0192	0	0
FFNN	Training	Acc	0.9851	1	1
FFNN	Validation	Loss	4.3784	10.7683	7.1403
FFNN	Validation	Acc	0.2727	0.1818	0.3636
CNN	Training	Loss	0.1173	0.0753	0.1017
CNN	Training	Acc	0.9403	0.9701	0.9552
CNN	Validation	Loss	3.2232	3.5301	4.2814
CNN	Validation	Acc	0.3636	0.3182	0.3182

TABLE II
1 layer, 50 filters, 50 neurons, filters of length 8, 16 and 24

Net	Set	Metric	Expt. 1	Expt. 2	Expt. 3
FFNN	Training	Loss	0	0	0
FFNN	Training	Acc	1	1	1
FFNN	Validation	Loss	8.9976	6.3715	7.4063
FFNN	Validation	Acc	0.3182	0.2273	0.3182
CNN	Training	Loss	0	0.0188	0
CNN	Training	Acc	1	0.9701	1
CNN	Validation	Loss	11.1176	7.1512	13.5941
CNN	Validation	Acc	0.3182	0.1818	0.2273

TABLE III
1 layer, 100 filters, 100 neurons, filters of length 8, 16 and 24

Net	Set	Metric	Expt. 1	Expt. 2	Expt. 3
FFNN	Training	Loss	0	0	0
FFNN	Training	Acc	1	1	1
FFNN	Validation	Loss	6.2946	5.7037	6.0173
FFNN	Validation	Acc	0.4091	0.2727	0.4091
CNN	Training	Loss	0	0	0
CNN	Training	Acc	1	1	1
CNN	Validation	Loss	13.3315	14.8280	12.3779
CNN	Validation	Acc	0.2273	0.2273	0.2727

TABLE IV
1 layer, 200 filters, 200 neurons, filters of length 8, 16 and 24

Net	Set	Metric	Expt. 1	Expt. 2	Expt. 3
FFNN	Training	Loss	0	0	0
FFNN	Training	Acc	1	1	1
FFNN	Validation	Loss	6.2173	3.9490	5.7697
FFNN	Validation	Acc	0.3636	0.5	0.4091
CNN	Training	Loss	0	0	0
CNN	Training	Acc	1	1	1
CNN	Validation	Loss	13.4876	15.0199	14.1734
CNN	Validation	Acc	0.2727	0.2273	0.2727

The first set of tests targeted the number of neurons. Big differences were tried and examined in order to find out if it is worth to use for example hundreds of them. From the result above, it is obvious that the network is functional. It is certainly able to predict outputs on given inputs and to train using the prepared generator. As the accuracy of training set predictions is tending to 1 in every case, it also seems that the network is learning well and can be used to classify such input data. So it can be used as a diagnostics support of musculoskeletal diseases. This particular result is crucial. However, it is also evident, that the classification of validation set was not satisfactory in any case.

The feed-forward neural network results are very varied. Despite the fact that with 200 neurons in every input layer's hidden layer, as well as in the main hidden layer, the network achieved a half accuracy, it was found out by further investigation, that the net simply learned to evaluate most of inputs with mark 1, which is the expected evaluation in almost half of samples. This way, the network found a coherence, which leads to good results, but in a totally improper way. Considering this, the feed-forward neural network was not providing a relevant comparison to the convolutional neural network and was omitted in further experiments.

The convolutional neural network supplies slightly more stable results. Furthermore, with increasing neurons count, its accuracy was decreasing, in contrast to the feed-forward neural network, which is interesting. As no other parameter has been changed and inverse tendency has been seen on the other network, it clearly is not connected with input layer's hidden layer and it is definitely caused by number of filters in the main hidden layer. Thus, consequent experiments used just a few convolutional filters and many neurons in fully connected layers. The next set of experiments addressed the impact of filters length to the prediction quality.

TABLE V
1 layer, 16 filters, 100 neurons, filters of length 8, 16 and 24

Net	Set	Metric	Expt. 1	Expt. 2	Expt. 3
CNN	Training	Loss	0.0581	0.0271	0.0482
CNN	Training	Acc	0.9701	0.9851	0.9701
CNN	Validation	Loss	4.9434	6.8729	5.5106
CNN	Validation	Acc	0.4545	0.4545	0.3182

TABLE VI
1 layer, 16 filters, 100 neurons, filters of length 4, 8 and 12

Net	Set	Metric	Expt. 1	Expt. 2	Expt. 3
CNN	Training	Loss	0	0.0193	0.0192
CNN	Training	Acc	1	0.9701	0.9851
CNN	Validation	Loss	12.3504	6.2219	6.5905
CNN	Validation	Acc	0.1364	0.3636	0.4091

TABLE VII
1 layer, 16 filters, 100 neurons, filters of length 16, 32 and 48

Net	Set	Metric	Expt. 1	Expt. 2	Expt. 3
CNN	Training	Loss	0.0187	0.0453	0.0190
CNN	Training	Acc	0.9701	0.9552	0.9851
CNN	Validation	Loss	9.2765	7.7833	7.8863
CNN	Validation	Acc	0.1818	0.0455	0.4091

An obvious improvement was done by lowering the number of filters. On the other hand, reduction, as well

as extension of filter length lowered the accuracy, so filter length was determined to be 8, 16 and 24.

Even so, these results are still not usable practically. The accuracy is not even 0.5. Another way to potentially improve the prediction, is to add more hidden layers.

TABLE VIII
2 LAYERS, 16 FILTERS, 100 NEURONS

Net	Set	Metric	Expt. 1	Expt. 2	Expt. 3
CNN	Training	Loss	0	0	0
CNN	Training	Acc	1	1	1
CNN	Validation	Loss	5.5643	6.1475	6.9884
CNN	Validation	Acc	0.5	0.4091	0.4091

TABLE IX
2 LAYERS, 16 FILTERS, TWO UNION FF LAYERS

Net	Set	Metric	Expt. 1	Expt. 2	Expt. 3
CNN	Training	Loss	0	0	0
CNN	Training	Acc	1	1	1
CNN	Validation	Loss	7.7642	8.1129	9.0588
CNN	Validation	Acc	0.3636	0.4091	0.2727

TABLE X
2 LAYERS, 16 FILTERS, TWO INPUT INNER FF LAYERS

Net	Set	Metric	Expt. 1	Expt. 2	Expt. 3
CNN	Training	Loss	0	0	0
CNN	Training	Acc	1	1	1
CNN	Validation	Loss	7.8389	6.7058	8.1535
CNN	Validation	Acc	0.3182	0.2727	0.3182

By adding another convolutional layer, the prediction has improved even more. However, adding another fully connected layer gave no indication of any improvement at all. Although the prediction was better, it cannot be considered reliable. Probable cause of this was simply a lack of provided samples.

Even though there were no other patterns to train on, it is possible to simplify the task to make classification easier. For this purpose, let us assume that patients are to be divided to just two groups; healthy ones and others. A healthy patient shall be a patient, evaluated by mark 0 or 1, while patients with higher mark shall form the other group. Like that, the network had only two classes to distinguish, and these two classes even contained about a half of the patients each.

To perform this simplified classification, the most successful network so far was used.

TABLE XI
SIMPLIFIED CLASSIFICATION

Net	Set	Metric	Expt. 1	Expt. 2	Expt. 3
CNN	Training	Loss	0	0	0
CNN	Training	Acc	1	1	1
CNN	Validation	Loss	3.8725	3.2748	2.9729
CNN	Validation	Acc	0.5	0.5909	0.6364

With only two (balanced) classes, the classification finally exceeded a 0.5 accuracy. However, by further investigation, it was found out that the predictions were more or less random. It was obvious that the higher accuracy is just a result of almost half-to-half distribution of patients in classes. That bit more than a half shows some success, but it is not nearly enough the network predictions as being trustworthy. There were apparently not enough input data to find correlations reliably.

V. CONCLUSIONS

In this article, there has been designed and implemented a diagnostic support tool of musculoskeletal diseases. The particular tool has been tested on real checkup data of patients with post-operative complications after a vestibular schwannoma surgery.

A basic definition of the tumor was provided, as well as its symptoms, impacts and current diagnostics. Then, artificial neural networks were introduced as a way of classification. The way of data collecting and its preprocessing have been described, too. Based on analysis of these data, a particular application has been implemented, resulting in a tool which predicts evaluation of patient's condition by movement of his or her face. Unfortunately, it has turned out that the predictions are not accurate yet. Hence, the network is not reliable.

However, according to its ability to learn the training samples, it seems that the developed convolutional neural network can find correlations in input data. And, thus, it indicates the potential to classify patients, as soon as enough data will be available to train on, which is just a matter of time.

Such an application could help both medical practitioners and patients. It opens the opportunity of self-diagnostics without the help of a doctor, possibly even from home, using just a device equipped with a stereoscopic camera. It also reduces the effect of a doctor's subjective view. In addition, there could be a mini-computer with a stereoscopic camera in a doctor's office, which could provide real-time evaluation of a patient.

Of course, vestibular schwannoma is not the only subject of body motion evaluation. The tool could be adjusted to another disease, which is diagnosed similarly.

REFERENCES

[1] National Institute on Deafness and Other Communication Disorders [online], "Vestibular schwannoma (acoustic neuroma) and neurofibromatosis." [Online]. Available: https://www.nidcd.nih.gov//health//vestibular-schwannoma-acoustic-neuroma-and-neurofibromatosis

[2] J. Kohout, J. Crha, K. Trnková, K. Štícha, J. Mareš, and M. Chovanec, "Robot-based image analysis for evaluating rehabilitation after brain surgery," *Mendel*, vol. 24, no. 1, pp. 159–164, 2018.

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

[4] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[5] S. Saha, "A comprehensive guide to convolutional neural networks — the eli5 way [online]." [Online]. Available: https://towardsdatascience.com//a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[6] M. Hüsken and P. Stagge, "Recurrent neural networks for time series classification," *Neurocomputing*, vol. 50, pp. 223–235, 2003.