Self Detection of Parkinson's Symptoms through machine learning

2018250042정현재

Abstract

Parkinson's disease is a typical neurodegen erative brain disease, which main symptom is the motor disorders. This disease is mainly caused by a lack of dopamine, which no longer regenerates after death. Parkinson's disease is not easy to distinguish from aging, so many patients overlook the symptoms and often come to the hospital after most dopamine dies and the disease progresses considerably. If the patients detect it early and find a disease before dopamine dies, it can delay symptoms. It can also significantly reduce the cost of treatment and time to support, and the nation's social support. Therefore, we use machine learning and deep learning technology so that patients with Parkinson's disease can easily check their condition.

Three major symptoms of Parkinson's disease, Bradykinesia, Hand Tremoring and Voice Problem, can be used to determine whether it is positive or not. Voice Problem and Bradykinesia utilize machine learning techniques with Binary Classification, which distinguishes Healthy/PD with Tabular data. Hand Tremoring uses Binary Classification based on the patient's spiral and wave drawing image. This utilizes the Convolutional Neural Network(CNN) in that it is image data. Through this, we want to make 'PD Symptoms Self Detection Model' that can easily determine suspected Parkinson's disease at home.

In the machine learning tasks, Decision Tree, Random Forest, KNN, SVM, and XGboost were adopted. Random Forest performed best on Voice data, and XGboost performed best on Hand Tremoring data. Random Forest and XGboost are ensemble models. KNN performed worst in both data.

In the CNN task, GoogleNet, VGG19, ResNet50 were adopted. Among them, GoogleNet performed best and ResNet50 performed worst.

1 Introduction

1.1 Statement, motivation, objective

In 2025, Korea will be expected to become a super-aged society, with the proportion of people 65 years or older exceeding 20%. As a result, medical expenses for the elderly population jumped from 10.8% (1990) to 40.8% (2018). The number of Neurodegenerative disease patients, which accounts for a significant portion of elderly disease patients, is increasing as the elderly population increases. Especially, the number of patients with Parkinson's disease, which causes movement disorders, has increased 2.5 times from 39,265 in 2004 to 96,499 in 2016. Also, the social costs of Neurodegenerative diseases are reported to be incomparably greater than other diseases. This consumes more direct and indirect costs than the costs for all the major causes of death, which is including cancer, cerebrovascular and cardiovascular diseases, and spends 8.7 trillion won a year on dementia-related costs including Alzheimer's and Parkinson's disease.[1] [2]

[1] Current status and future of Parkinson's disease in Korea, The Korean Movement Disorder Society

[2] Passive tracking of dyskinesia/tremor symptoms, United States patent applicaion

Despite these serious social problems, diagnosis research of Parkinson's disease progresses slowly because until recently we understood that dopamine cells died out, causing Parkinson's disease. Even if diagnoses Parkinson's disease early, we concluded that dopamine cells have already died so we could not prevent the progression of the disease and just improved movement disorders by taking **medication. However, a recent study found that** dopamine cells fall asleep before they died, and even in these conditions, we can show the symptoms of movement disorders of Parkinson's disease. In other words, abnormal activation of the astrocyte in the basal ganglia and substantia nigra regions causes excessive transmission of GABA(γ-aminobutyric acid), which is the inhibitory neurotransmitter, and it makes dopamine cells fall asleep. It gives us some chances that if we can diagnose Parkinson's disease before the dopamine cells die out, we might wake up the sleeping dopamine cells to prevent Parkinson's disease. So we have looked at ways to self-diagnose Parkinson's disease and it helps to prevent the progress of disease and to save on the social costs of this by inducing Parkinson's disease patients to visit the hospital early and get diagnosed thoroughly.

The biggest problem with diagnosing Parkinson's disease is that the patients are not easily aware of the initial symptoms. The delay in visiting the hospital makes treatment difficult. A survey of 857 patients and protectors of Parkinson's disease at major university hospitals across the country found that it took an average of 9.4 months to visit the hospital after Parkinson's disease symptoms occurred. One out of every four patients (26%) were found to visit the hospital a year after the symptoms occurred, while another study found that 17% of all patients took more than five years to visit the hospital after the symptoms. After the symptoms occurred, 23% (113 people), 25% (123 people) of the family, and 52% (254 people) of the patients did not know about Parkinson's disease, and about half of the patients did not suspect Parkinson's disease even if they had Parkinson's disease symptoms. Older patients are alert to Alzheimer's disease which causes cognitive impairment, but they cannot alert to Parkinson's disease which movement disorder proceeds slowly because of mistaking it for the symptom of aging.

To make the patients with Parkinson's disease visit the hospital on time, we would like to present a solution called the Model which self-diagnose symptoms of Parkinson's disease. It is easy to measure at home and it can help identify suspicious symptoms that the patients did not know. Also, the patients can consistently check their condition in real-time and it makes the progression of the disease slow. To do this, we will identify the major symptoms of Parkinson's disease and a test that can be measured at home. After that, we analyze the symptoms by using artificial intelligence. It can help the patients of Parkinson's disease in the economic, social, and medical fields, and also placing great significance on the entire society of the healthcare system and medical artificial intelligence.

2 Datasets

Parkinson's disease occurs due to the deterioration of the black-matter in the midbrain. The midbrain is the source of Dopamine, the critical neurotransmitter for movement. As the midbrain degenerates, neurons can't receive enough amount of Dopamine and it causes neurons to stop moving slowly so several physical symptoms are following.

Bradykinesia, tremor of limbs, speech changes are the most common symptoms of Parkinson's disease. We will analyze the correlation between the results of the diagnosis and the data of symptoms. Specifically, we will create an self-diagnosis model via supervised learning,

and find the correlation between the features via unsupervised learning.

We will target some suspicious symptoms that can be easily diagnosed at home. Hand trembling, voice, walking habits are good examples. It is more convenient because patients don't use heavy machines that need to be used in hospitals such as MRI, CT, and PET. With a simple device, patients will be able to diagnose their symptoms personally.

We will use three types of data. They are images, voice, table, respectively. We were able to get these data from the Michael J. Fox Foundation[3], PPMI(The Parkinson's Progression Markers Initiative)[4], under the permission to access and use of the data. We also fetched some other data from Kaggle.com.[5]

3 Model

3.1 Voice data

3.1.1 Data Architecture

The overall behavior of people suffering from bradykinesia is gradually slowing down. Since this also can be the symptom of getting older, it is hard to conceive this as an initial symptom of Parkinson's. We have two different data for this.

[3] The Michael J.Fox Foundation, 'michaeljfox.org', Parkinson's Research Agency

[4] Parkinson's Progression Markers Initiative, 'ppmi-info.org', non-profit organization which is being sponsored by The Michael J.Fox Foundation. 'Motor Assessments Gait Data & Arm Swings' dataset. permitted by PPMI

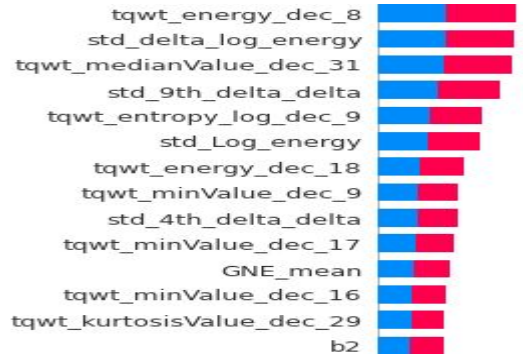
[5] Kaggle. machine learning competitions and now also offers a public data platform, 'www.kaggle.com'. 'Parkinson's Drawings', 'Parkinson's Voice Data' Open Dataset

The first one is voice data as a CSV file. This consists of 758 rows of the number of people and 750 columns of features of voice. People are either patient(1) or normal(0), and the representative features are Maximum vocal fundamental frequency, Minimum vocal fundamental frequency, Amplitude, Frequency variation, etc.

3.1.2 Data Preprocessing

Although it does not have missing values, there is a problem that there are 756 features. It consists of many similar features when we check this with 'Seaborn heatmap'. However, since the number is so huge, we decided to use the library 'Shap'[6] instead of manually weeding similar features out.

Shap automatically checks the level of contribution of each feature on learning and align features by that. This requires a fitted model, so we applied Shap on every model we used. It means important features are different for each model.



There is one more problem. 3 rows have the same id, and an id can be distributed both in train and test set. In this case, the model would learn the similarity of the features that comes from the same id and it could boost the performance unintentionally. We set 10-fold cross validation by id to prevent that situation.

3.1.2 Five Adopted Models

[6] https://github.com/slundberg/shap

1. Decision tree - To learn a model based on rules according to data uniformity. Methods for determining data uniformity include gini impurity and information gain. The deeper the tree is, the greater the risk of overfitting, so the predictive performance of the decision tree is likely to be degraded. Therefore, the lower tree under a specific node of the decision tree formed by the maximum tree should be pruned to enhance the generalization performance

2. KNN - It is an algorithm that finds K data close to the current data within the existing data to classify the current data as a specific value and classifies the current data with the most classified values.

3. SVM - SVM is an algorithm that seeks to find the best boundary to classify classes. Margin is the shortest distance from observations to hyperplane and the SVM wants to maximize this margin when classifying. In other words, it is a technique to find boundaries that can maximize margins. SVM is distinguished from linear and nonlinear, and first, linear SVM is an algorithm that makes the decision boundary linear. Nonlinear SVM has a nonlinear decision boundary. By adding characteristics, we can create a linearly delimited dataset, and typical methods include adding a multinomial kernel or adding similarity characteristics. This allows the application of kernel tricks to achieve the same results as adding characteristics without actually adding polynomial properties, or to add characteristics by measuring how much each sample resembles a particular landmark. The similarity function to be measured can then define Gaussian radial basis function (RBF). The parameters of the RBF include gamma and C. The smaller the gamma value, the wider the bell-shaped graph, and the smoother the decision boundary because the sample affects a wider range. C is the cost, and the lower the value, the higher the margin and the higher the learning error rate, the more the decision boundary is created. If C is too low, there is a risk of under-fit, and if too high, there is a risk of over-fit, so we should find the appropriate value.

4. Ensemble - Ensemble is said to be one of the most popular methods of classification, and is favored to show high performance in the predictive analysis area of structured data. It is generally divided into bagging and booting methods. In the case of bagging, each classifier takes different data sampling based on the same type of algorithm and performs the botting by performing the learning. The method of sampling and extracting data from individual classifiers is called the bootstrapping segmentation method. Unlike cross-validation does not allow overlays between data sets, the method of bagging permits overlaying. Boosting is learning and forecasting by giving weights to the next classifier so that several classifiers can perform learning sequentially but the classifiers learned earlier can correctly predict data that is not predicted correctly.

4-1. RandomForest classification - It is an algorithm that makes multiple classifiers using a tree algorithm to make a final decision by boating as a representative ensemble model of the method of bagging. Parameters include variables that specify the number of decision trees called n_estimators and parameters used to improve over-fitting in the decision tree, such as max_feature, max_depth, min_samples_leaf.

4-2. XGboost - It is a model that adds hyperparameters to regulate overfitting in GBM and parameters that can be stopped early. It can reduce the number of divisions by pruning divisions that no longer have positive gains, thereby reducing the possibility of causing too many divisions, thus preventing overfit. In addition, there is an early interruption function that allows it to stop repeats in the middle if the evaluation value of the evaluation data set is optimized through cross-validation rather than the specified number of iterations. At this time, GBM is a typical algorithm that performs bushing while weighting error data, and is a

model that uses Gradient descent to update weights.

3.1.3 Reasons for selecting each model

First, we chose the decision tree because it is easy to visualize the model made regardless of the scale of the feature. Knn was chosen because it was easy and intuitive to understand and was a good classification method for large numbers of samples. Svm is a popular model in machine learning and has the advantage of being well suited to complex classification problems, so it was chosen and returned. When we selected the svm kernel, we selected the linear, rbf kernel, and when we tried, the linear's performance was much lower than that of the rbf, so we chose the rbf to train the model. The performance seems to be better when using a nonlinear kernel because the dataset is complex and difficult to categorize linearly. Finally, we selected xgb, a model that complements the shortcomings of gbm, among the typical bagging methods, rfc and booting.

3.1.4 Hyperparameter Tuning

Before we trained the model, we did GridSearch with pre-processed dataset and important features came out from Shap.

Hyperparameters tested and the outcomes are shown respectively from here. They can be easily distinguished by color.

1 Decision Tree

'max_depth', 'max_leaf_nodes', 'min_samples_split', 'min_impurity_decrease', 'min_samples_leaf', was given a range of up to [1~100], [1~100], [1~1000], [0~1], [1~1000].

{max_depth=None, max_leaf_nodes=3, min_impurity_decrease=0.0, min_impurity_split=None, min_samples_leaf=1}.

2 RandomForestClassifier

'n_estimators' :[1,5,10,50,100,200], 'max_depth' : [2,3,4,5,6,8,10,12], 'min_samples_leaf': [2,3,4,5,6,7,8,10], 'min_samples_split' : [2,3,5,6,7,8,9,10

{'max_depth': 12, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 200}.

3 KNN

{'n_neighbors' :[3, 4, 5, 6, 7], 'leaf_size' : [20, 30, 40, 50]}

{'leaf_size': 20, 'n_neighbors': 5}

4 SVM

{'gamma':['scale', 'auto', 0.001, 0.01, 0.1, 1, 10, 25, 50, 100], 'break_ties':[True, False]}
{'break_ties': True, 'gamma': 'scale'}

5 XGboost

{'n_estimators' : [1,10,20,50,100,200], 'learning_rate' : [0.01,0.02,0.03,0.05,0.1,0.15,0.2], 'max_depth' : [2,3,4,5,6,7,8]}
{'learning_rate': 0.01, 'max_depth': 6, 'n_estimators': 100}

3.1.5 Model comparison

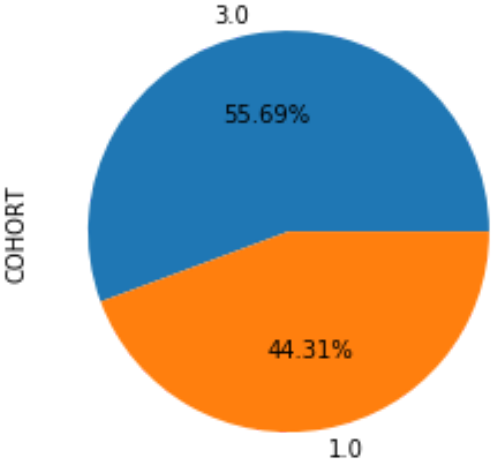
With hyperparameters got from the last section, we fitted each model and compared the AUC scores of them.

Model	AUC score
Decision Tree	0.6948
Random Forest	0.7456
KNN	0.6655
SVM	0.7108
XGboost	0.7104

Random Forest got the best score, and KNN got the worst. We assume that Random Forest showed its strength of combining multiple models. However, KNN showed its weakness that possibility of getting biased model from the local with skewed data.

3.2 Arm swing data

3.2.1 Data Architecture



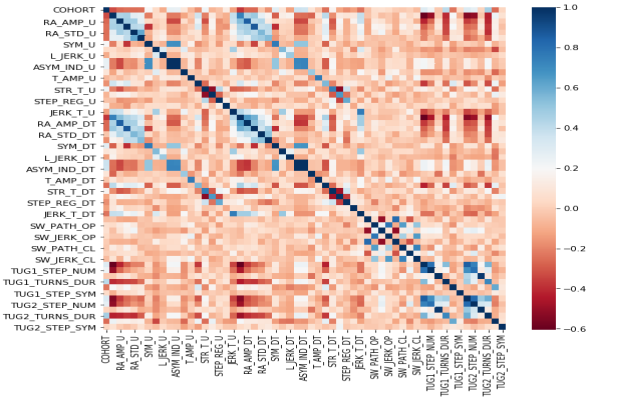
Gait_data is the walking data analyzed to enable systematic study of motion. It is used to measure physical exercise, physical dynamics, muscle activity, etc, and is used to evaluate and treat individuals with conditions that affect walking ability using walking analysis. A more sensitive test of motor function is needed to increase the likelihood of ascertaining a kinetic change, and it is possible to measure and evaluate the progress of diagnosis and disease through quantitative measurement of gait and mobility.

Gait_data consists of 60 columns and 193 rows. Among the columns, PATNO, EVENT_ID and INFODT are the patient's ID, and COHORT is a label which means 1-asymptomatic relatives, 2-true controls, 3 - PD. The table below shows the distribution of the data. The distribution ratio of COHORT is shown in the figure.

3.2.2 Data Preprocessing

Typical encoding methods include label encoding and one hot encoding, but data shows that all features are numeric values that are not categorical, so encoding is not necessary. However, because column 'COHORT' is a label, it is better to change numeric type to category type.

Handling of missing values is very important and we decided to handle this about two methods. First delete missing value and second is to replace(imputate) missing value with median. Changing to a median is better than



changing to an average.

Adjusting the range of values of different variables to a constant level is called feature scaling.

- standardization - Each feature in the data is converted to a value with a Gaussian normal distribution with a mean of zero and a variance of one. Each feature has a different range of values, so if convert them to values between 0 and 1, we can compare them all in the same size units.
- minmaxscaler - If the distribution of the data is not Gaussian, this can be applied.

Remove characteristics that have information that is not useful in the work and remove features that have the least impact.

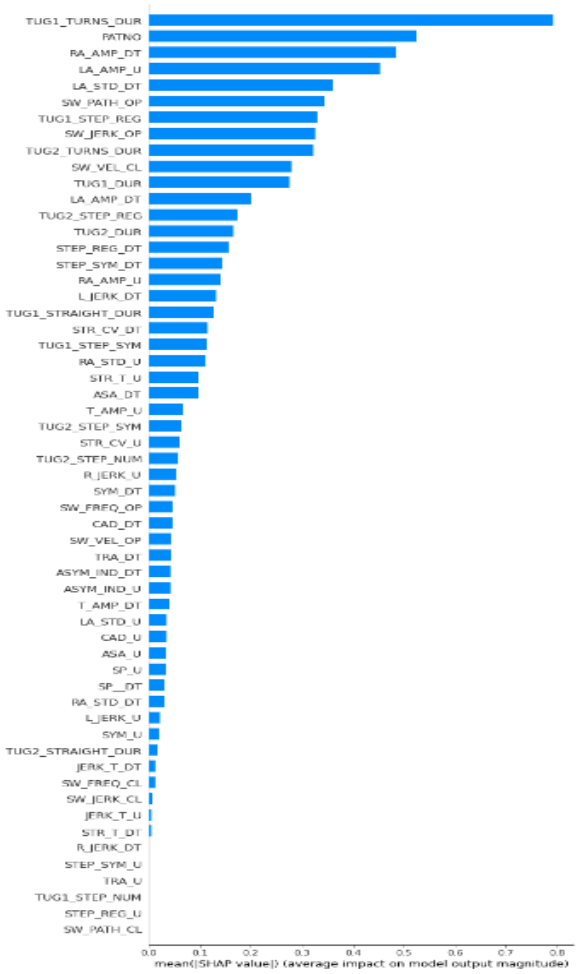
Outlier is a value with an anomaly value that deviates from the pattern of the overall data. We applied the Inter Quantile Range (IQR) method to find and remove outliers. IQR can be visualized in a box plot manner using a technique that utilizes the deviation of the Quantile value. This method usually uses the range generated by multiplying IQR by 1.5 to determine the maximum and maximum values, and then considers the data above or below the maximum values as ideal.

When preprocessing gait data, we have to

remove individual information assigned to the patient from the feature first because of avoiding confusing. After this, we removed the outliers. In the correlation heatmap, the higher the positive correlation, the closer the dark blue, and the higher the negative correlation, the closer the red.

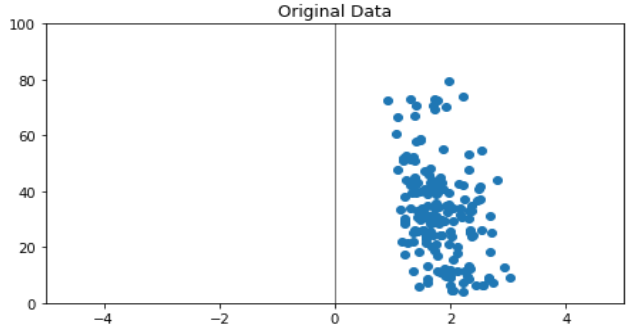
Using the get_outlier function, we found and deleted the outliers in each column. After removing outliers, we have to decide whether to remove the NaN value first or replace it with the median value. To this end, put in the data with the NaN value removed first in the Data_drop_first, and Data is replaced with the missing value to the median value. After fitting several models with two datasets, we adopt a more accurate dataset.

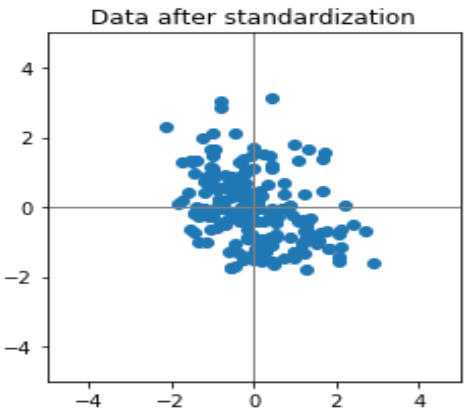
To extract features, we trained Data_drop_first dataset through ridge model. The cross validation technique was used to prevent overfitting. Also, to set the best parameter, train the model by writing a function which finds the alpha value. Also, remove features that do not affect performance by using SHAP. Looking at the table below, the features of 'SW_PATH_CL','STEP_REG_U','TUG1_STEP_NUM','TRA_U','STEP_SYM_U','R_JERK_DT' are deleted from the feature because they do not affect the model.



Normalize the data after completing the feature extraction. Also, we removed the missing value instead of replacing. The table below shows data_drop_first dataset.

After normalization, the distribution of TUG1_TURNS_DUR, RA_AMP_U, which have a significant effect on the feature was plotted. In the orginal data, the range of y-values was jagged, but after normalization, the minimum value of each column is changed to 0 and the maximum value to 1.





There is an another way to train datasets that changed all nan values to median before removing missing values. Because we changed all the nan values, the nan values disappear and the number of rolls comes out 193 for each column.

To sum up, we preprocess gait_data by eliminating anomalies, handling nan values, normalizing, and feature extraction, and we are going to choose between deleting all data containing nan first or changing nan to median.

3.2.3 Model design

The most accurate model will be selected among models made using data that have pre-removed missing values and data that have changed missing values to median. When training models, we set test_set to 0.1 and preceeding train_test_split. To prevent overfitting, cross_validation was also applied, and cv = 10 was set. AUC will be used to determine the accuracy of the model as described above.

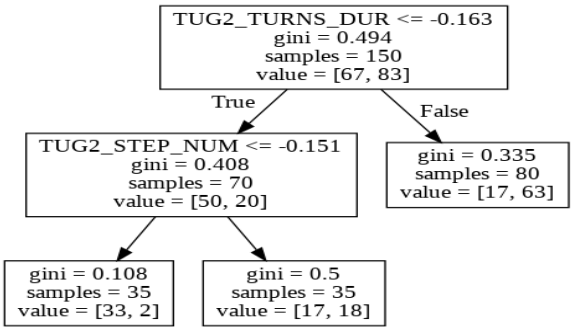
We trained the model with the pre-processed dataset and used decision tree, randomforest, knn, svm and xgboost like we did on the voice data. The current report will focus on the accuracy values calculated using different models and the resulting model selection, and the final report will analyze the criteria for selecting models when fitting them, why they chose final models, and why they could have good accuracy. Before analyzing the model, let me introduce some tools for analyzing accuracy.

1) Data which remove missing values first

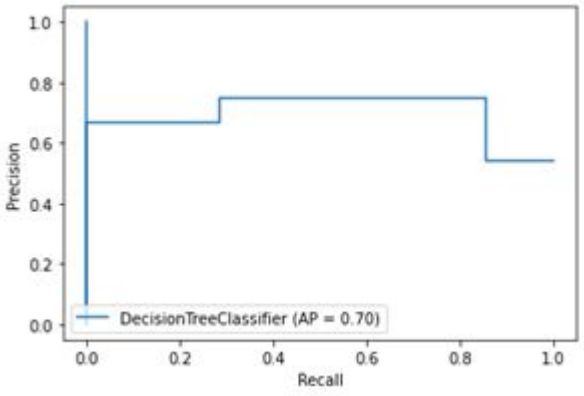
① Decision Tree

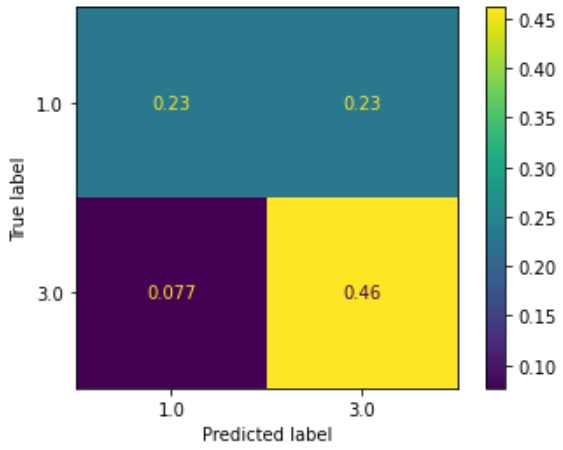
In order to find the best parameters, the range for each parameter was set and the most accurate parameter was selected. Each parameter, 'max_depth', 'max_leaf_nodes', 'min_samples_split', 'min_impurity_decrease', 'min_samples_leaf', was given a range of up to [1~100], [1~100], [1~1000], [0~1], [1~1000] and the parameter with the highest accuracy was selected. The best parameter is {min_samples_leaf=1, min_samples_split=35}. The accuracy was `0.767948717948718` as a result of cross_validation.

Figure visualizes the crystal tree of the optimal Decision Tree model.



To analyze the accuracy, the confusion matrix and the precision_recall curve are shown in Figures.

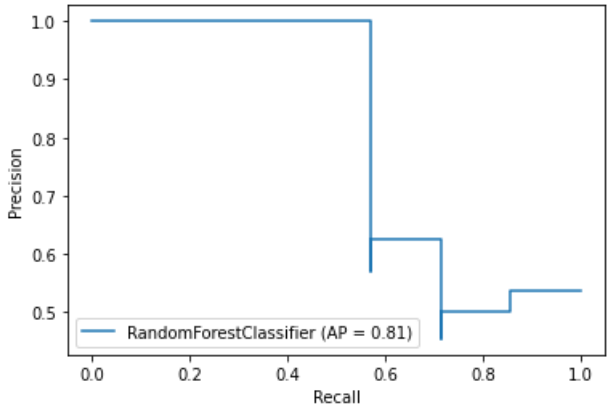




Finally, roc_auc_score is `0.7023809523809524`.

② RandomForestClassifier

Gridsearch was executed to find the best parameter. The parameter range was set to 'n_estimators' :[1,5,10,50,100,200],'max_depth' : [2,3,4,5,6,8,10,12],'min_samples_leaf': [2,3,4, 5,6,7,8,10],'min_samples_split' : [2,3,5,6,7,8,9,1 0] and gridsearch was executed, and the best_param was {'max_depth': 6, 'min_samples_leaf': 3,'min_samples_split': 8,'n_estimators': 10} and the accuracy is `0.7584980237154151`.

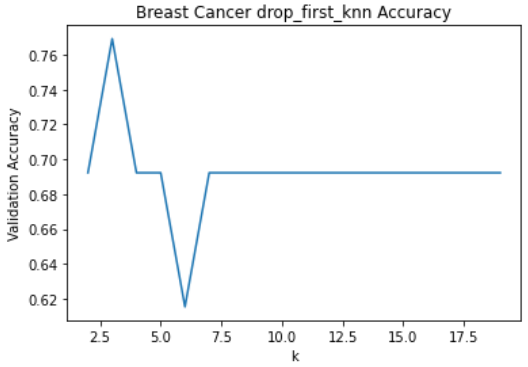


Finally, roc_auc_score is `0.7142857142857143`.

③ knn

To identify the best n_neighbors parameter of knn, the accuracy was expressed in the graph shown in Figure 2-6 below after turning it all from 2 to 20. The best core is 0.7692307692307693, when n_neighbors = 3, and when cv was returned, the accuracy was `0.7692307692307693`

Finally, roc_auc_score is `0.7692307692307693`.



④ svm

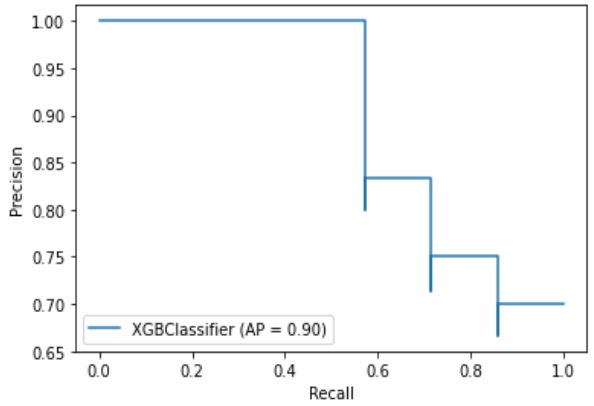
The svm model has two methods. The first is a linear method and the second is a non-line method. First, we proceeded with the linear model and the accuracy was 0.6166666666666667 but the non-linear was 0.6878787878787879 so found the best parameter for this. The parameter range was set to {'C': [0.001, 0.01, 0.1, 1, 10, 25, 50, 100],'gamma':[0.001, 0.01, 0.1, 1, 10, 25, 50, 100] }and gridsearch was executed, and the best_param was { C=1, gamma=0.01} and the accuracy is `0.7288461538461538`

Finally, roc_auc_score is `0.7380952380952381`

⑤ XGBoost

The parameter range was set to {'n_estimators' : [1,10,20,50,100,200], 'learning_rate' : [0.01,0.02,0.03,0.05,0.1,0.15,0.2], 'max_depth' : [2,3,4]}

and gridsearch was executed, and the best_param was {'learning_rate': 0.15, 'max_depth': 2, 'n_estimators': 10} and the accuracy is 0.7448717948717949.



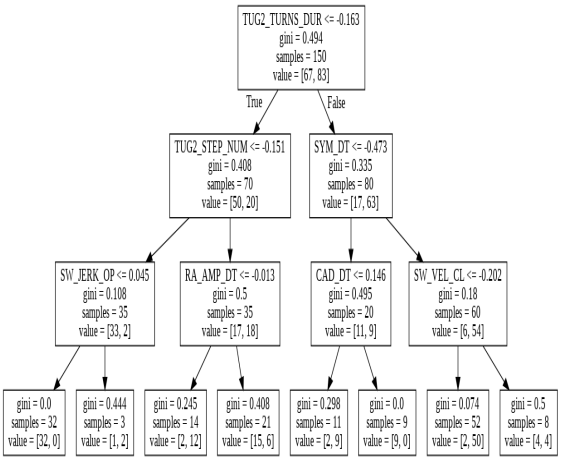
Finally, roc_auc_score is 0.8571428571428572.

2) Data that changed missing values into median values

① Decision Tree

In order to find the best parameters, the range for each parameter was set and the most accurate parameter was selected. Each parameter, 'max_depth', 'max_leaf_nodes', 'min_samples_split', 'min_impurity_decrease', 'min_samples_leaf', was given a range of up to [1~100], [1~100], [1~1000], [0~1], [1~1000] and the parameter with the highest accuracy was selected. The best parameter is {min_samples_leaf=1, min_samples_split=2, max_depth=3}. The accuracy was 0.7235294117647059 as a result of cross_validation.

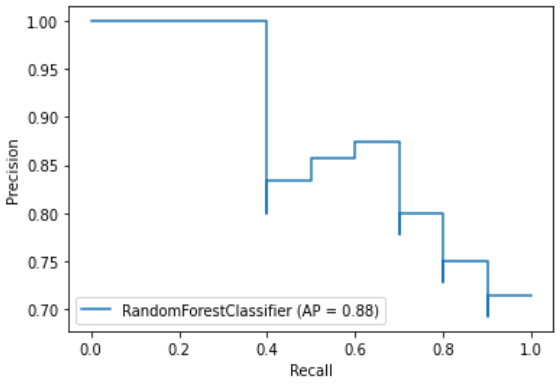
Figure visualizes the crystal tree of the optimal Decision Tree model.



Finally, roc_auc_score is 0.7910714285742871.

② RandomForestClassifier

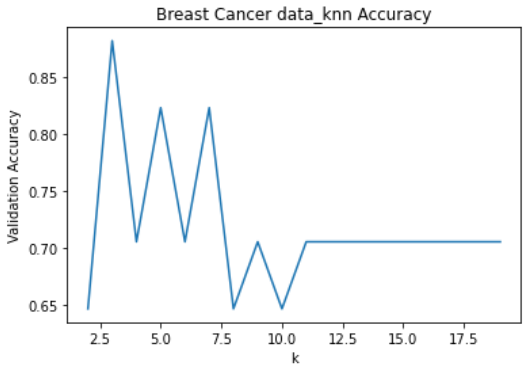
Gridsearch was executed to find the best parameter. The parameter range was set to 'n_estimators' :[1,5,10,50,100,200],'max_depth' : [2,3,4,5,6,8,10,12],'min_samples_leaf': [2,3,4,5,6,7,8,10],'min_samples_split' : [2,3,5,6,7,8,9,10] and gridsearch was executed, and the best_param was {'max_depth': 6, 'min_samples_leaf': 3,'min_samples_split': 8,'n_estimators': 10} and the accuracy is 0.7533333333333333.



Finally, roc_auc_score is 0.8285714285714285.

③ knn

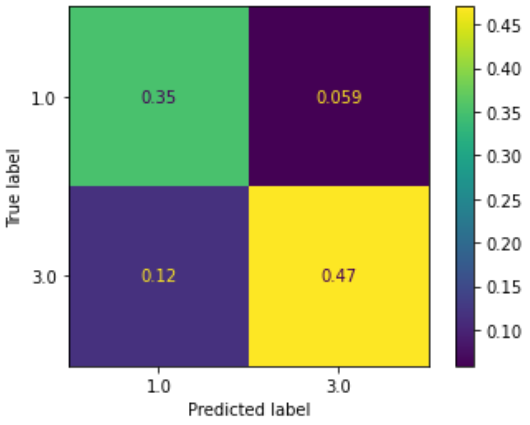
To identify the best n_neighbors parameter of knn, the accuracy was expressed in the graph shown in Figure 2-18 below after turning it all from 2 to 20. The best core is 0.8823529411764706, when n_neighbors = 3, and when cv was returned, the accuracy was 0.64.



Finally, roc_auc_score is `0.7857142857142857`.

④svm

The svm model has two methods. The first is a linear method and the second is a non-line method. First, we proceeded with the linear model and the accuracy was 0.6166666666666667 but the non-linear was 0.728846153846153 so found the best parameter for this. The parameter range was set to {'C': [0.001, 0.01, 0.1, 1, 10, 25, 50, 100],'gamma':[0.001, 0.01, 0.1, 1, 10, 25, 50, 100] }and gridsearch was executed, and the best_param was { C=1, gamma=0.01} and the accuracy is `0.7733333333333333`.



Finally, roc_auc_score is `0.8142857142857143`

⑤ XGBoost

The parameter range was set to {'n_estimators' : [1,10,20,50,100,200], 'learning_rate' : [0.01,0.02,0.03,0.05,0.1,0.15,0.2], 'max_depth' : [2,3,4]}

and gridsearch was executed, and the best_param was {'learning_rate': 0.15, 'max_depth': 3, 'n_estimators': 50} and the accuracy is `0.7647058823529411`.

Finally, roc_auc_score is `0.8857142857142858`

3.2.4 Result

The data_xgb has the highest accuracy when representing values for a total of 10 models with roc_arc_score. This has shown that xgb models perform best and that datasets that change missing values to median perform better than those that pre-remove missing values. Also, data_xgb was the best performance even when looking at acuracy instead of auc. Final report will cover the analysis.

	roc_auc_score
data_xgb_auc	0.885714
drop_first_xgb_auc	0.857143
data_rfc_auc	0.828571
data_svm_auc	0.814286
data_dt_auc	0.791071
data_knn_auc	0.785714
drop_first_svm_auc	0.738095
drop_first_rfc_auc	0.714286
drop_first_dt_auc	0.702381
drop_first_knn_auc	0.700872

	accuracy
drop_first_dt_auc	0.782692
data_svm_auc	0.773333
data_xgb_auc	0.764706
drop_first_rfc_auc	0.759684
data_rfc_auc	0.753333
drop_first_xgb_auc	0.744872
data_dt_auc	0.718015
drop_first_knn_auc	0.706061
drop_first_svm_auc	0.687879
data_knn_auc	0.640000

3.3 Parkinson's Drawings

3.3.1 Architecture

This data is hand-drawn data from patients with Parkinson's disease and normal people. There are spiral and wave in Drawing datasets. Spiral has 51 normal people data, 51 Parkinson patient data, and the wave file also has 51 normal data and 51 Parkinson patient data. Data type is image so we can choose CNN model to train model. Frist of all, we proceeded with preprocessing since the dataset was not refining. To do this, we use pytorch because the CUDA engine used by the pytorch can produce enormous computational speeds, and there are many relatively simple and easy-to-use functions such as forward() and backward().

3.3.2 Data preprocessing

we created a csv file with Image name, type, and normal/patient group for the column first. This is to easily determine the name and type in the future. After that, when writing code, we built a model separately by separating spiral and wave.

Other than that, we changed normal label to 0 and patient label to 1. we used read_csv for csv file and cv2 for image file. Below are the images for csv file and spiral/wave drawings.

CNN especially needs many additional processes. First, we need to resize the image and convert it to a tensor. So we brought the image to cv2, adjusted the size, and made the SpiralData class, WaveData class, which converts it to the tensor value. Also we used the transforms of phytorch's torchvision. To increase data with basic adjustments, we used RandomHorizontal Flip, which randomly rotates images left and right reversal. After that, we transformed the data using SpiralData class and WaveData class, and tied them to Dataset at once. And then process train_test_split using random_split from torch.utils. The ratio was set at 8:2, and we tied it up by 64.

Not only the number of models in CNN is huge, but also the structures and trends of models vary frequently. So among the

image-classification algorithms, we've selected one that showed the best performance at the recent contest, and that is 'VGG19'. It is the model of the Oxford-team VGG, who won second place in the Imagenet visual recognition contest in 2014. Even if the team GoogLeNet got the championship, VGG19 is much simple than the algorithm of GoogLeNet. That's why we choose VGG19.

Firstly, the layer was set to be 19. ReLU was adopted and the return value was activated by softmax. The loss function is CrossEntropyLoss, the optimizer is Adam of Param. We are about to set the diverse value of learning_rate from 0.001 to 0.1. Eventually, num_epoch was set to be 4.

Hand tremoring is a very significant symptom of Parkinson's. That is the reason why researchers often use hand drawing in diagnosing and rehabilitation of the disease. There is also a paper [7] that explains the enormous influence of drawing when it comes to diagnosing Parkinson's. This data describes the drawings of patients and normals. The difference between them is apparent. The data contains 102 spiral drawings and 102 wave images. The proportions of the two categories are the same. Since there is no matched label for each drawing, and this is so complicated, we converted this to a CSV file. After converting, as you can see below, spiral data has 'HE' and wave data has 'HO' & 'PO'. Each of them is placed under the 'NAME' label. Under the preprocessing, there were two obstacles to be considered. The first one is related to the shape of each image, and the second one is about the size of the data.

1. The margins of each image are big. They do not have any RGB values, and we don't see any way to apply some weights. So we didn't apply any preprocessing on images.

2. The size of data is small, and there is an obvious pattern in there. Since this can be the cause of overfitting, we applied some augmentations such as rotating and flipping images. We also utilized RandomHorizontalFlip in the transform procedure.

After dealing with these problems, we created Class and Transform to learn CNN. The images are converted to 'torch tensor' by the Transform function, which is in DataLoader. In the end, the converted images are bound into batch_size.

3.3.3 Model design

Under the selection of CNN model, we had to decide on some hyperparameters.

The higher the learning rate, the sharply it goes down. So we need to choose an adequate level of learning rate. We used 0.1, 0.01, 0.001 and 0.0001 for that.

The loss function means the way that lowering the error rate. There are diverse loss functions. By utilizing some of them, we finally choose CrossEntropyLoss because it showed the highest accuracy.

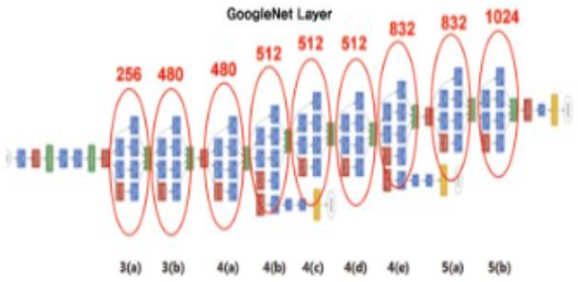
The optimizer stabilizes learning speed and makes it fast. Adam is the most recent optimizer so far, so we adopt it.

Here is the summary of the 3 models.

1. GoogleNet caught the champion on the Large Scale Image Recognization Challenge 2014. It consists of 22 layers and utilizing 'softmax' as its activation function. Additionally, it already showed the highest performance on Parkinson's drawing dataset.[8]

[7] Digitalized spiral drawing in Parkinson's Disease: a tool for evaluating beyond the written trace. Jérémy Danna, PhD Jean-Luc Velay, PhD,1 Alexandre Eusebio, MD, PhD, 2,3 Lauriane Véron-Delor, MSc, 1,4 Tatiana Witjas, MD, PhD, 2,3 Jean-Philippe Azulay, MD, PhD, 1,2 and Serge Pinto, PhD4

[8] Inception(Going Deeper with Convolutions) - Byung Kyu Kang. 'kangbk0120.github.io'

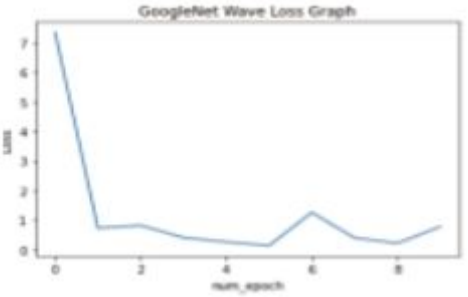


The GoogleNet model has two distinct features. First, the GoogleNet uses Inception model. This complements the shortcomings of Dropout, which makes data sparse and does not learn properly. It also solved the amount of computation and the shortcomings of the Deep Layer, which can cause Overfitting. The Insertion Module performs a 1x1, 3x3, 5x5 Convolution operation respectively to extract features efficiently. This also performs Max pooling of 3x3 and adds Padding, which is rare in the pooling operation, because the input and output H and W must be the same. As a result, the process of feature extraction was to maintain as sparse as possible, and the matrix operation was to combine them to make them dence as much as possible. All the red circles shown above are where the Inception module was used. In addition, additional pooling layers were inserted in, so it makes the size to be reduced. There is another additional feature, auxiliary classifier. That's not only at the very end, but in the middle of the process of extracting results through softmax. This can help solve the Vanishing Gradient, which disappears more than some value in the middle. Instead of the last calculation of Loss, the gradient can be adjusted properly, calculating Loss in the middle of each Inception Module.

Considering the characteristics of this model, we was set four hyperparameters.

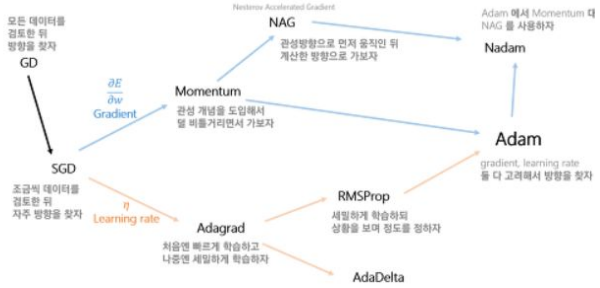
① learning_rate : We adjusted the model by setting [0.1, 0.01, 0.001, 0.0001] in turn. We wanted to add more numbers, but it was impossible because it took a long time. In the above numbers, we found that the value of 0.001 was the highest accuracy. We have confirmed that the proper value of leading_rate reduces loss most efficiently.

② num_epoch : We also adjusted num_epoch by setting [5, 10, 15]. However, as we can see from the loss graph, no further adjustments occurred from the moment the epoch value was above 3. So the values above 3 all showed similar performance.



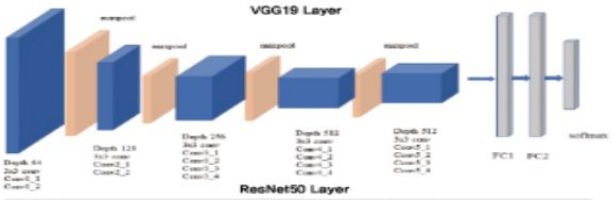
③ Loss_func : There are many different types of loss function. There are values such as SGD, Hinge, and Binary cross-entropy, which belong to simple types, and rather than SGD and Hinge, which have added simple regularization, we decided to use Binary Cross_entropy technology, which shows a high correlation to the surrounding values.

④ Optimizer : Optimizer is a model that adjusts learning speed quickly and reliably. Unlike other models, optimizer had many advanced technologies. Beyond the simple model, Adam is the latest generation model that controls the Gradient and leading rate.[9]



[9] Neural Network Part 8. Optimizer - Gom Guard's Library 18.03.29 'https://gomguard.tistory.com/187'

2. VGG19 yielded second place on the LSIRC2014. As shown in its name, it contains 22 layers and also uses softmax as the activation function. It generally shows a relatively high-performance rate, despite its simple structure.[10]



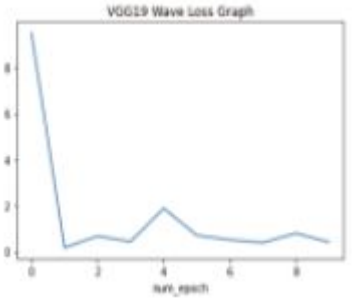
We observed that classification error declines with the depth getting deeper by 11, 13, 16, 19. This directly means that the performance is getting better, so we choose VGG19 which has the biggest layer among VGG models. In CNN, the higher the weight, the slower the learning speed exponentially. VGG has a relatively high speed since it has a low weight. We gradually increased the non-linearity of the feature by increasing the number of layers step by step.

① learning_rate

We adjusted the model by tuning the learning rate in the range of [0.1, 0.01, 0.001, 0.0001]. We wanted to add more numbers, but we could not because it required too much learning time. We found 0.001 as the best, and it is not so high nor so low. This is the same with GoogleNet.

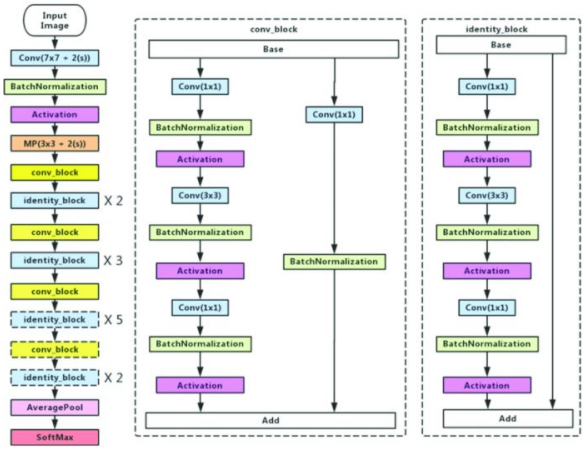
② num_epoch

The range of tested num_epoch was [5, 10, 15]. When it exceeded epoch 3, it stuck to a narrow range, so we expected the fluctuation of performance would be stable after the 3rd epoch.



We adjusted 'loss_function' and 'optimizer' as we did in the GoogleNet section, and the outcome also was similar to that.

3. ResNet50 has a helluva complicated structure. It has ReLU as its activation function and frequently uses activation on each layer.[11]



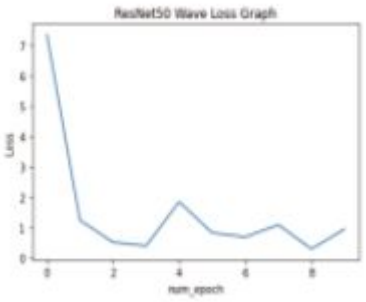
With the outcome of the last two models, we confirmed that the thicker the network, the higher the performance. In that sense, we wondered what it would be with ResNet50. Like the name, it has 50 layers and much heavier than the last two models.

We used Softmax as the activation function on the last two models, but this time, we used ReLU. Also, we applied a technique called 'Skip

[10] VGG-19 Architecture 19.06.28 'https://bskyvision.com/504'

[11] Convolutional Neural Network Architecture 3 Resnet - AI vision Raon People. Naver Blog 18.04. 23

connection' that directly connects the input of the layer to the output. As we were concerned about gradient vanishing due to the model's deep network, we evaluated the loss changing.



Fortunately, there was no gradient vanishing since the size of data wasn't so large and the number of layers was not that heavy. However, acceptable performance does not mean that's a good one since it showed some instability and the worst accuracy score. We assume that the number of layers has a limited correlation with high performance. When the number of layers is too high, it can lead to even lower performance.

3.3.4 Result

The summary of model training is below.

input -> forward -> loss check -> backward(backpropagation) -> optimizer -> repeated until num_epoch.

We put train data in the model and inspected the loss of that.

Approximately, we could find the similarity of loss variation from after 1st epoch. So we set the num_epoch as 6~10, which is the proper number for the task.

The accuracy scores of each model follow.

Accuracy of Test Data: 71.42857360839844 GoogleNet Spiral Data

Accuracy of Test Data: 66.66666412353516 VGG19 Spiral Data

Accuracy of Test Data: 57.14285659790039 ResNet50 Spiral Data

Accuracy of Test Data: 71.42857360839844 GoogleNet Wave Data

Accuracy of Test Data: 71.42857360839844 VGG19 Wave Data

Accuracy of Test Data: 61.904762268066406 ResNet50 Wave Data

The range of 66~71 accuracy was shown, and among them, the score of ResNet50 Spiral Data is the lowest.

4 Conclusion

4.1.1 Result analysis

In both data, ensemble models performed the best, which is better than a single model because the ensemble model is a model that improves predictability by combining different learning models. Generally used as the basic algorithm of an ensemble is a decision tree. The decision tree can be applied very easily and usefully, but it must have a complex rule structure to improve its predictive performance, which can result in overfitting and degrade its predictive performance. However, these shortcomings serve as advantages in ensemble. The ensemble combines learning algorithms with very low predictive performance to improve predictive performance while continuing to update probabilistic supplements and weights for areas where errors occurred, as the decision tree becomes a good weak learner. Knn predicts only through close neighbors, so the forecasts are more biased toward local information than other algorithms, which can lead to poor performance.

In the CNN task, we could make some assumptions based on the results. First, the number of layers should not be too high or too

low. Too few layers provoke low accuracy and too many layers can make loss value higher due to vanishing gradient. Therefore, we assume GoogleNet prevailed because its number of layers is 22, which between 19 of VGG19 and 50 of ResNet50. Also, other issues enhanced accuracy, such as the density of data was adjusted by the inception module or the auxiliary classifier technique was utilized.

We anticipate inspecting the correlation thoroughly and getting better performance with more data and GAN.

4.1.2 Further Expectation

Through various symptoms, we have implemented a total of three models to determine whether or not having Parkinson's disease. One unfortunate thing was that because each data came from different sources, it was impossible to combine the three data into a single model because each patient had different symptoms. Therefore, when predicting Parkinson's disease through suspicious symptoms, we put each symptom in each model and concluded that if any of the three results are positive, the patients should suspect Parkinson's disease. If we can measure all three symptoms of a patient with Parkinson's disease, we can combine the models with these data. Then we can compare the performance of the combined models with the performance of the three models we have created in our current project to conclude what is more accurate.