Early detection of Cognitive Impairments: AI approach[⋆]

Yevgeniya Daineko [1,†], Tamerlan Egemberdiev[1,*,†] and Zhibek Zholdasova [2,†]

[1] *International Information Technology University, Manas st 34/1 050040 Almaty, Kazakhstan*
[2] *Center for the Treatment of Neuroses and Alzheimer's Disease, Almaty, Republic of Kazakhstan*

Abstract

Early detection of cognitive impairments, such as Mild Cognitive Impairment (MCI) and Alzheimer's Disease (AD), is crucial for timely intervention and improved patient outcomes. This study evaluates the effectiveness of the Leap Motion Controller (LMC) in assessing cognitive function through hand movement analysis, utilizing motion tracking and machine learning techniques. A total of 93 participants, including individuals with MCI, AD, and normal cognition, were assessed using the "CogniQuest" application, a Unity-based system integrating digitized neuropsychological tests—Clock Drawing Test (CDT), Trail Making Test (TMT), and Bells Test—with LMC technology. Machine learning models, including Support Vector Machines (SVM), Logistic Regression (LR), and Convolutional Neural Networks (CNNs), were employed to classify cognitive status based on motion-tracking biomarkers. The system achieved an 88.5% accuracy rate, with a 40% correlation between hand movement patterns and neuropsychological test outcomes, outperforming traditional cognitive assessments. The results highlight the potential of LMC-based motion analysis as a non-invasive, cost-effective diagnostic tool for early cognitive impairment detection. Future research will focus on dataset expansion, model refinement, and broader healthcare integration.

Keywords

cognitive impairment, mild cognitive impairment, Alzheimer's disease, motion tracking, machine learning, neuropsychological testing, early diagnosis

1. Introduction

The term "cognitive impairment" refers to a variety of disabilities or restrictions in cognitive function that make it difficult for the person to process information, stay focused on their work, retain important details, and go about their daily lives [1]. In addition to various mental processes that form the basis of the cognitive functions includes memory, attention, problem solving, language comprehension and decision-making. If any of these processes do not occur normally, people may struggle to deal with the problems that arise during their daily tasks. They range from mild to severe. Traumatic brain injury, chronic illnesses and other neurodegeneration disorders that could lead to these kinds of diseases are among the common and important root causes. Moreover, long-term issues such as substance abuse, sleeplessness, and chronic stress further aggravate this deteriorating process of cognition. Understanding this phenomenon necessitates action for improving life quality of people with cognitive deficit. These deficits can be minimized through interventions such as drugs, rehabilitation therapy, and lifestyles adjustments to enhance adaptable cognitive functioning.

Cognitive impairments, including Alzheimer's disease, dementia, and other forms of cognitive decline, are an increasing global concern. The Alzheimer's Disease International (ADI) has actively worked for over a decade to raise awareness about dementia worldwide. Awareness plays a crucial role in how societies perceive and address this condition, influencing public policies, healthcare strategies, and the quality of care provided to those affected. Every three seconds, a new case of dementia is diagnosed somewhere in the world. As of 2019, approximately 55 million people globally were living with dementia, and according to WHO projections, this number could rise to 139 million by 2050. The economic burden is also increasing; in 2019, the global annual cost of

[⋆] *AIT 2025: 1st International Workshop on Application of Immersive Technology, March 5, 2025, Almaty Kazakhstan*
[1*] Corresponding author.
[†] These authors contributed equally.

y.daineko@iitu.edu.kz (Y. Daineko); timugin007@gmail.com (T. Egemberdiev); zhibek_zholdas@mail.ru (Zh. Zholdas)
0000-0001-6581-2622 (Y. Daineko)

dementia care reached $1.3 trillion, and by 2030, it is expected to exceed $2.8 trillion. With an aging global population, dementia is becoming a leading cause of mortality, underscoring the need for early detection and intervention to mitigate its impact on patients, families, and healthcare systems. Recent reviews emphasize the growing role of artificial intelligence in this field. For instance, [2] provides a detailed overview of AI-based diagnostic techniques—from neuroimaging to sensor data analysis—for Alzheimer's disease, demonstrating the potential of these methods to support early, non-invasive diagnosis and decision-making

Kazakhstan faces similar challenges in addressing cognitive impairments. Official statistics from 2020 indicate that approximately 145 individuals were registered with Alzheimer's disease in the country [3]. However, experts suggest that the actual number is significantly higher, with estimates exceeding 200,000 based on international healthcare trends [4]. Cognitive impairments can remain in a latent state for 10-15 years before becoming symptomatic, making early detection critical [5]. Without timely intervention, patients lose essential abilities such as mobility, speech, and self-care, increasing their dependence on caregivers and healthcare institutions. While treatment options are currently limited, advances in technology offer promising new approaches for identifying cognitive impairments at an early stage.

Recent advancements in neurophysiology and digital health have led to the development of new diagnostic tools utilizing motion-tracking technology [6]. Research suggests a strong correlation between upper limb motility and cognitive state [7,8]. Leap Motion Controller (LMC) technology has emerged as a promising solution, allowing for non-invasive, real-time tracking of hand and finger movements. Compared to conventional diagnostic methods such as MRI and neuropsychological assessments, LMC is cost-effective, easy to implement, and accessible in both clinical and home environments. This project explores the potential of LMC combined with machine learning algorithms to assess cognitive status accurately and facilitate early intervention.

2. Theoretical part

Before analyzing specific tests, it is essential to distinguish between neuropsychological and cognitive assessments. Neuropsychological tests evaluate brain function and behavior relationships, linking specific cognitive functions to particular brain structures [7]. Cognitive tests, in contrast, assess general cognitive abilities like memory, attention, language, and reasoning but cannot independently diagnose cognitive disorders [8]. They help identify potential cognitive issues that require further medical evaluation.

Clock Drawing Test (CDT)

The CDT assesses cognitive function by requiring individuals to draw a clock with a specific time. It evaluates executive function, visual-spatial skills, motor programming, attention, and concentration [9]. The test is widely used for early dementia screening, as difficulties in clock drawing often indicate cognitive decline. Unlike language-based tests, CDT is less influenced by education level and cultural background, making it more universally applicable. Scoring varies across systems, but standard evaluation involves analyzing number placement, hand positioning, and symmetry [10].

Trail Making Test (TMT)

Developed in 1944, TMT measures visual attention, task switching, and processing speed. The test has two parts: TMT-A requires connecting numbered circles sequentially, while TMT-B alternates between numbers and letters. It assesses executive function, with longer completion times indicating cognitive impairment [10]. The test is highly sensitive to detecting early cognitive decline and frontal lobe dysfunction [11].

Bells Test

The Bells Test evaluates visual neglect and attention by asking individuals to identify target symbols among distractors. It is particularly useful for assessing spatial neglect in stroke patients [12]. Scoring is based on accuracy and omissions, helping detect cognitive deficits affecting perception and attention.

3. Correlation between hand motor abilities and cognitive functions

Aging is often accompanied by a decline in visual-motor skills, with research showing that individuals with mild cognitive impairment (MCI) exhibit more significant deficits than healthy older adults [13]. Hand movement impairments have been linked to early stages of dementia, reinforcing the potential for motor-based assessments in detecting cognitive decline. Studies indicate that individuals with MCI and Alzheimer's Disease (AD) demonstrate slower, less coordinated hand motions, suggesting that visuomotor impairments precede advanced motor symptoms.

Finger motor abilities, including dexterity and movement precision, have been correlated with cognitive performance. Tests measuring parameters such as movement amplitude, velocity, acceleration, and rhythm have shown significant differences between MCI/AD patients and healthy individuals [7]. The Spearman correlation analysis further confirms that specific hand movement features are linked to cognitive impairment severity. These findings support the application of motion-tracking technology like LMC in diagnosing cognitive decline through hand function analysis.

4. Exploring using of LMC in cognitive health

Most Leap Motion Controller (LMC) research focuses on therapy and rehabilitation, with limited studies on its use for cognitive illness diagnosis. A Google Scholar and PubMed search using "Leap Motion Controller" and "Cognitive" found only 12 relevant studies, with just one directly addressing diagnostics [15].

A systematic review following PRISMA guidelines identified 19 peer-reviewed studies on LMC in psychological areas such as ADHD, ASD, dementia, and MCI. These studies primarily used game-based interventions, improving motor skills, attention, and socialization. Another study examined the effects of structured exercise (SE) and LMC-based play therapy (LMCBET) on cognitive function and quality of life in older adults, showing positive outcomes.

Overall, existing research highlights LMC's potential in therapy and rehabilitation, emphasizing the need for further exploration in cognitive diagnostics.

5. Machine learning algorithms

Machine learning (ML) enables computer systems to learn from data, improving performance over time. ML algorithms analyze patterns in historical data to make predictions, classify information, and assist in decision-making [16]. In this study, three classifiers—Logistic Regression (LR), Support Vector Machine (SVM), and Convolutional Neural Network (CNN)—are used to develop a predictive model for cognitive impairment detection:

- SVM detects subtle movement changes linked to cognitive decline.
- LR quantifies the relationship between motor function and cognitive scores.
- CNN extracts key visual features from hand movement data for accurate classification.

By integrating these classifiers, the system enhances diagnostic precision, aiding early cognitive impairment detection. A similar approach has previously been used to predict cardiovascular disease, demonstrating the versatility of machine learning in medical diagnostics [17]. However, as AI becomes embedded in clinical workflows, ensuring the security and privacy of medical data is paramount. In [18] it was discussed the architecture and implementation of secure AI-driven diagnostic platforms, emphasizing encryption, secure data access, and compliance with healthcare regulations.

6. Results and discussion

Software design and structure

The design phase plays a critical role in developing a system that identifies people at risk of cognitive impairment based on Leap Motion data. At this stage, the structure, functions, and interactions of the main components are determined. The design process defines the system architecture, user interface, data processing algorithms, diagnostic integration strategies, and testing and validation paths. The importance of this stage is obvious from the establishment of the basic principles and directions of development that ensure the efficiency and successful implementation of subsequent stages of the project.

In this diagram (Fig.1), LMC captures the movements of hands and fingers, the data is processed using the selected machine learning stack, and the results are stored in the SQLite database. Unity integrates with the Leap Motion SDK and uses the Ultra Lean plugin to track hand movements, and the user interface is designed and prototyped using figma.

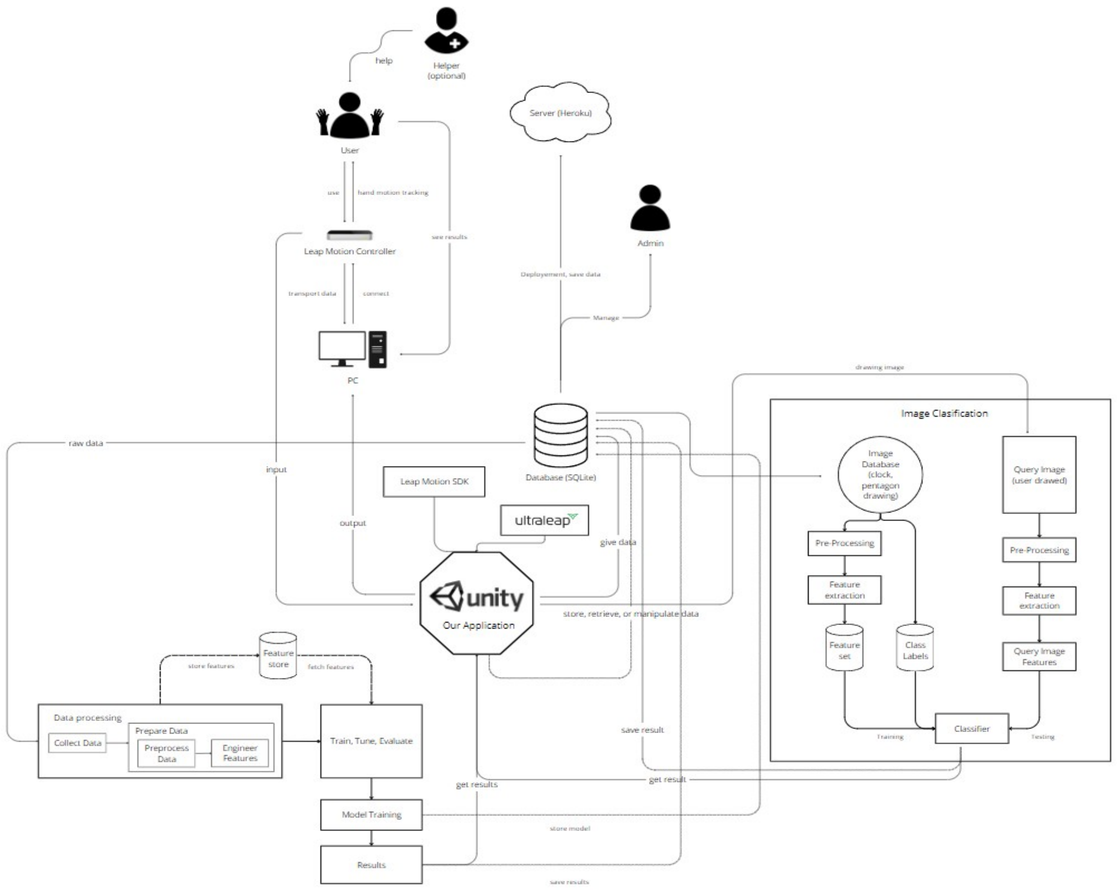


Figure 1: Physical model

Here's a detailed explanation of the diagram's components and their interactions:

- User and Helper Interaction: A user interacts with the system by performing hand gestures, which are captured by the LMC. An optional helper can assist the user if needed.
- Leap Motion Controller: The LMC tracks hand and finger movements in real-time. These movements are captured as raw data and transmitted to a PC via a USB connection.
- PC and Leap Motion SDK: The PC processes the raw data using the Leap Motion SDK, which converts the raw hand motion data into a usable format. This processed data is then integrated with Unity for further processing and visualization.
- Unity Integration: Unity, a game development platform, integrates with the Leap Motion SDK through the UltraLeap plugin. This integration allows Unity to utilize the hand motion data for various applications, such as controlling a user interface or interacting

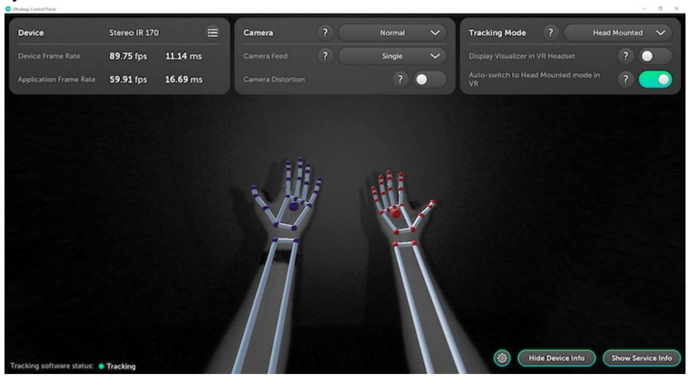
with virtual objects. The user interface for the application is designed and prototyped using Figma, a collaborative interface design tool, ensuring a user-friendly experience.

- Database (SQLite): The SQLite database is used to store, retrieve, and manipulate data. The database holds various types of information, including raw hand motion data, processed results, and image classification data.
- Image Classification Module: Image database stores various images for classification purposes, and images are pre-processed to enhance their quality and suitability for feature extraction. Key features from the images are extracted and stored in a feature set. A classifier is trained using the extracted features and class labels. The classifier can then be used to predict the class of new query images based on their extracted features.
- Data Processing and Model Training: The data processing pipeline involves collecting data, preparing it through preprocessing and feature engineering, and storing the features for later use. The collected data is used to train, tune, and evaluate machine learning models. The results of the model training are stored and can be retrieved for analysis and further improvements.
- Results and User Feedback: The final results, whether from hand motion tracking or image classification, are stored in the database. Users can view these results through the Unity application, which provides a seamless and interactive experience.

Moreover, this diagram showcases a comprehensive system that captures hand and finger movements using LMC, processes the data using machine learning techniques, stores the results in an SQLite database, and integrates with Unity for an interactive user experience. The image classification module further extends the system's capabilities by enabling the classification of images based on extracted features and trained classifiers.

Experimental setup

To ensure proper functionality of the Leap Motion device, certain conditions depicted in picture (Fig. 2). These conditions include factors such as adequate lighting, minimal obstructions in the device's field of view, and appropriate positioning of the device relative to the hands being tracked. Additionally, is essential to ensure that the Leap Motion software is correctly installed and configured on the computer or system where it will be used. Regular calibration and software updates also contribute to optimal performance and verifying that the USB connection is secure can prevent connectivity issues.



Device: Stereo IR 170
Device Frame Rate: 89.75 fps 11.14 ms
Application Frame Rate: 59.91 fps 16.69 ms
Camera: Normal
Camera Feed: Single
Camera Distortion
Tracking Mode: Head Mounted
Display Visualizer in VR Headset
Auto-switch to Head Mounted mode in VR
Tracking software status: Tracking

Figure 2: Hands tracking software

The controller tracks hand movements within an interactive 3D zone with a preferred depth of 60 cm (23 inches) to a maximum of 80 cm (31 inches), extending beyond the device's field of view of 140 x 120°. It's important to ensure that subjects do not experience discomfort while using LMC. The photo below illustrates the optimal position of the device and the recommended hand placement level (Fig.3).

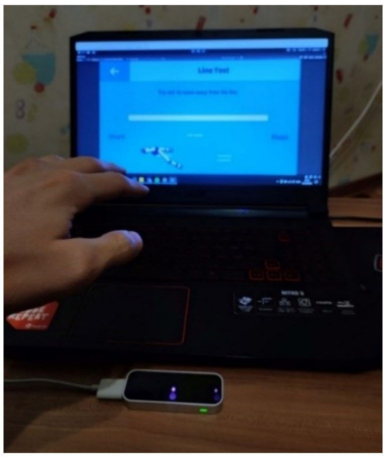


Figure 3: Installation of using LMC

Machine learning pipeline

In Jupiter Lab, TensorFlow serves as the primary framework for developing a model to classify images. To begin, crucial libraries are imported, such as TensorFlow, NumPy for numerical computing, Seaborn for data visualization, and other necessary modules for preprocessing and analysis. This ensures access to the tools needed for building and evaluating the image classification model effectively.

Image Classification model

As outlined in the theoretical section, the MNIST dataset containing handwritten numbers for drawing a clock is required for evaluation. Data cleanliness is crucial at this stage as it directly impacts the accuracy of the model and the quality of predictions it generates. Upon inspection, the dataset consists of 60,000 images distributed across 10 classes in the training dataset and 9,895 images distributed similarly in the testing dataset. The data is then normalized to ensure consistency and optimize model performance.

Processing the image using the Image Data Generator class from Keras. image_size = (40, 40) size of the images is 40 pixels. batch_size = 64, which means that 64 images will be processed during each training iteration. The Image_generation object uses random scaling of 0.5–1.0 times compared to their original size, arbitrary brightness adjustment from 0.2 to 1.0. flip images horizontally with a 50% probability. val_gen = defines 99% of the data that will be used for verification. These augmented images will be used to train the CNN model, providing more diverse data to better generalize the model.

Then load the data and process it using DirectoryIterator. For the Training Set, variable generation, termed image_generation, is used. No changes are made to the images in the Test set; only 99% of the images are taken. class_mode='categorical': The mode for class labels. In this case, the classes are presented in a categorical format, which means that the labels are vectors with a single encoding. seed=1337: This sets a random initial value for the reproducibility of the dataset. subset='training': indicates that it is used for a training subset of the dataset. shuffle=False: disables shuffling of the dataset, that is, the order of the verification images and their corresponding labels will be preserved. A list of classes and the number is also kept.

Here is a histogram (Fig. 4) that illustrates the number of images by class in the training dataset. As well as some processed images from the training set, with changed brightness and with scaling.

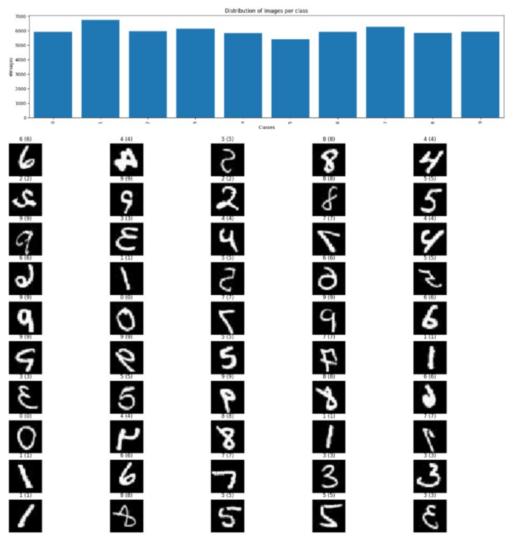
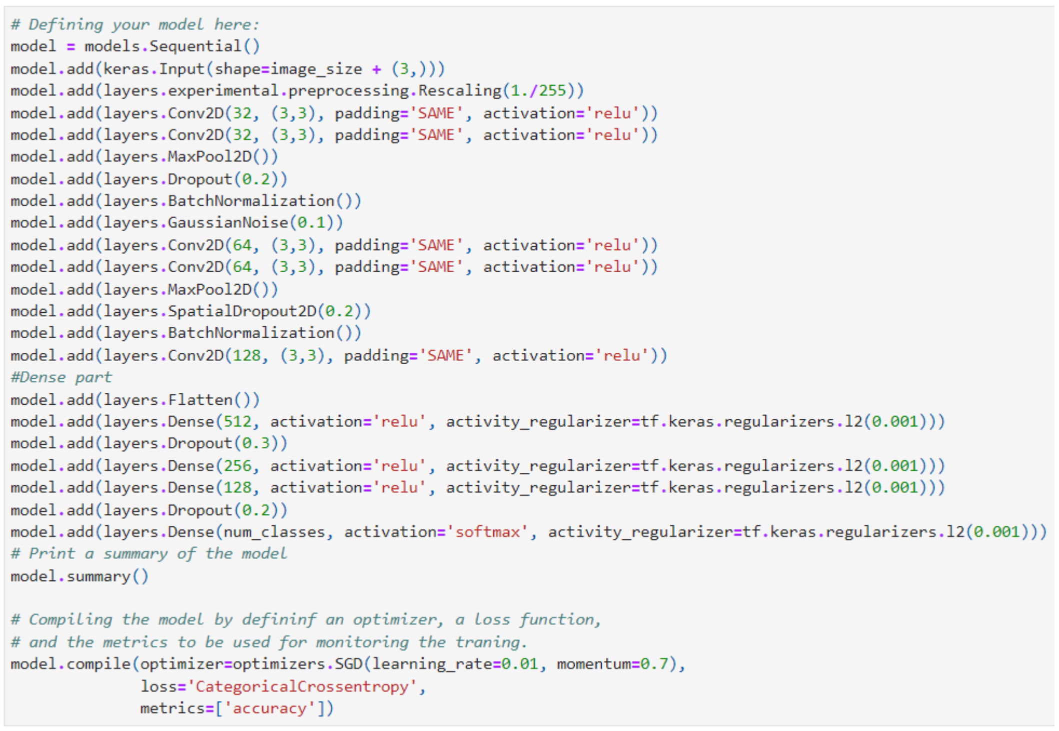


Figure 4: Histogram and number of images by class

The architecture of CNN is presented here (Fig. 5). He designed a CNN model with multiple convolutional layers followed by several fully connected layers. And in the provided code snippet, the CNN architecture is developed. At the beginning, the model is created by Sequential(). Then adding the size of the image and three channels (RGB). Then the input values are scaled by dividing them by 255.



```
# Defining your model here:
model = models.Sequential()
model.add(keras.Input(shape=image_size + (3,)))
model.add(layers.experimental.preprocessing.Rescaling(1./255))
model.add(layers.Conv2D(32, (3,3), padding='SAME', activation='relu'))
model.add(layers.Conv2D(32, (3,3), padding='SAME', activation='relu'))
model.add(layers.MaxPool2D())
model.add(layers.Dropout(0.2))
model.add(layers.BatchNormalization())
model.add(layers.GaussianNoise(0.1))
model.add(layers.Conv2D(64, (3,3), padding='SAME', activation='relu'))
model.add(layers.Conv2D(64, (3,3), padding='SAME', activation='relu'))
model.add(layers.MaxPool2D())
model.add(layers.SpatialDropout2D(0.2))
model.add(layers.BatchNormalization())
model.add(layers.Conv2D(128, (3,3), padding='SAME', activation='relu'))
#Dense part
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu', activity_regularizer=tf.keras.regularizers.l2(0.001)))
model.add(layers.Dropout(0.3))
model.add(layers.Dense(256, activation='relu', activity_regularizer=tf.keras.regularizers.l2(0.001)))
model.add(layers.Dense(128, activation='relu', activity_regularizer=tf.keras.regularizers.l2(0.001)))
model.add(layers.Dropout(0.2))
model.add(layers.Dense(num_classes, activation='softmax', activity_regularizer=tf.keras.regularizers.l2(0.001)))
# Print a summary of the model
model.summary()

# Compiling the model by definінf an optimizer, a loss function,
# and the metrics to be used for monitoring the traning.
model.compile(optimizer=optimizers.SGD(learning_rate=0.01, momentum=0.7),
              loss='CategoricalCrossentropy',
              metrics=['accuracy'])
```

Figure 5: Architecture of CNN

Then see those CNN modules that were recently discussed. This is a Conv2D with 32 filters, filter size (3, 3), "SAME" filling and "relu" activation. The "SAME" filling ensures that the spatial dimensions of the output feature map match the input ones. Then another two-dimensional convolutional layer is added with the same characteristics as the previous one. MaxPool2D() adds a maximum merge layer that reduces the spatial dimensions of the feature map by taking the maximum value within each merge window. Dropout(0.2) adds a dropout layer with a dropout factor of 0.2, which helps prevent overfitting by randomly setting the proportion of input units to 0 during training. Batch Normalization() adds a batch normalization layer that normalizes the activations of the previous layer, improving the stability and convergence of the model during training.

GaussianNoise(0.1) adds a layer that applies Gaussian noise with a standard deviation of 0.1 to the input data. This can act as a form of regularization and helps the model to generalize better. Flatten() adds a smoothing layer to transform a 2D feature map into a 1D feature vector, preparing it for fully connected (dense) layers. Dense(512, activation='relu', activity_regularizer=tf.keras.regularizers.l2(0.001))) adds a dense layer with 512 units and 'relu' activation. It also applies L2 regularization with a regularization strength of 0.001. Dropout(0.3), Dense(256, 128) Additional dense layers are added with a decreasing number of units of measurement and regularization. Dense (num_classes, activation='softmax', activity_regularizer=tf.keras.regularizers.l2(0.001)) adds an output layer with num_classes units and 'softmax' activation, which is suitable for multiclass classification tasks. L2 regularization is also applied to the layer.

At the very end, it compiles a previously defined model using the specified optimizer, loss function, and metrics. Here is the explanation of the code:

model.compile(optimizer = optimizers.SGD (learning_rate=0.01, momentum=0.7), loss='CategoricalCrossentropy', metrics=['accuracy'])

Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.01 and a momentum of 0.7. The learning rate determines the step size when updating the model weights during training, and momentum helps speed up the optimization process by adding part of the previous weight update to the current update.

loss='CategoricalCrossentropy' defines the loss function to be used during training. Categorical crossentropy is a commonly used loss function for multiclass classification problems.

metrics=['accuracy'] Metrics are set to evaluate the performance of the model during training and evaluation. In this case, accuracy is chosen as an indicator, which calculates the proportion of correctly classified samples.

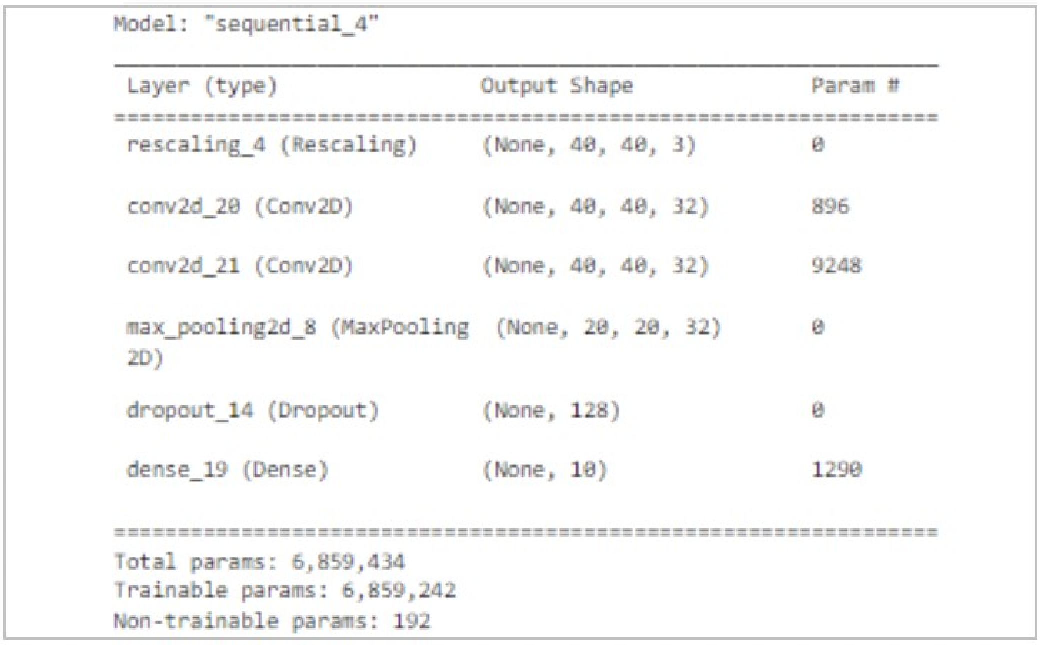
Three callbacks that are used during model training.

Early stop this callback tracks the specified metric (in this case, "loss") and stops the learning process ahead of time if the tracked metric does not improve over a certain number of epochs. This helps to prevent overfitting and reduces the amount of unnecessary calculations. The "Patience" parameter determines the number of periods without improvement after which training will be discontinued.

ReduceLROnPlateau This callback reduces the learning rate when the monitored metric (in this case, "loss") stops improving. This allows adjusting the learning rate more precisely during training to help reach a better decision. The factor parameter determines the coefficient by which the learning rate decreases, and the patience parameter indicates the number of periods without improvement after which the learning rate will be reduced. The min_lr parameter sets the lower limit of the learning rate.

ModelCheckpoint this callback saves the model with the best performance based on the specified metric (in this case 'val_accuracy'). This enables maintaining the weight of the model during training and using the most effective model for later use. The save_best_only parameter ensures that only the best model is saved, and the monitor parameter specifies the metric that will be monitored to determine the best model.

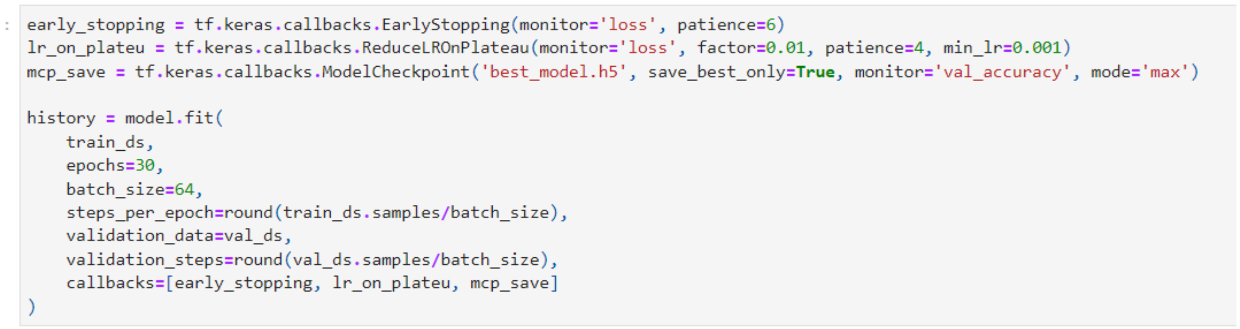
In addition to these callbacks, understanding the architecture of the model is vital. The architecture of a sequential model typically comprises layers with specific characteristics. Each layer serves a unique purpose and contributes to the overall structure of the model. The Output Form column displays the output form of each layer in the format (batch_size, height, width, channels). The Parameter # column shows the number of parameters to be trained in each layer. Untrained parameters display the number of untrained parameters in the model. The model contains a total of 6,859,434 parameters, and 6,859,242 of them are trainable in picture (Fig. 6). This distinction highlights the trainable parameters essential for model optimization through training iterations.



```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_4 (Rescaling)     (None, 40, 40, 3)         0

 conv2d_20 (Conv2D)          (None, 40, 40, 32)        896

 conv2d_21 (Conv2D)          (None, 40, 40, 32)        9248

 max_pooling2d_8 (MaxPooling  (None, 20, 20, 32)       0
 2D)

 dropout_14 (Dropout)        (None, 128)               0

 dense_19 (Dense)            (None, 10)                1290

=================================================================
Total params: 6,859,434
Trainable params: 6,859,242
Non-trainable params: 192
```

Figure 6: Description of the model

Then the model (Fig. 7) is trained using the fit() function and sets various parameters and callbacks. The epoch determines how many times the entire training dataset will be passed through the model during training. In this case, the model will be trained for 50 epochs. batch_size defines the number of samples for each gradient update. steps_per_epoch defines the number of steps (packets) to be processed in each epoch. validation_data is used to evaluate the performance of the model on a separate validation dataset during training. validation_steps define the number of steps (packets) to be processed from the validation dataset at each epoch.



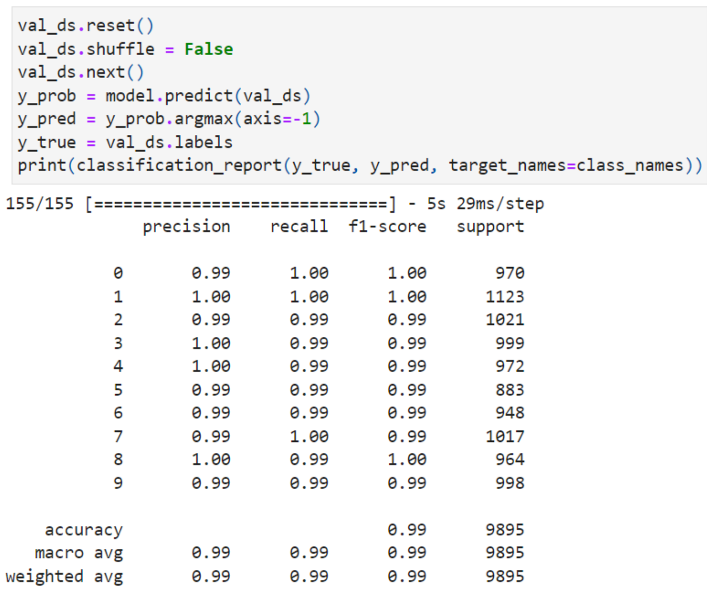
```
early_stopping = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=6)
lr_on_plateu = tf.keras.callbacks.ReduceLROnPlateau(monitor='loss', factor=0.01, patience=4, min_lr=0.001)
mcp_save = tf.keras.callbacks.ModelCheckpoint('best_model.h5', save_best_only=True, monitor='val_accuracy', mode='max')

history = model.fit(
    train_ds,
    epochs=30,
    batch_size=64,
    steps_per_epoch=round(train_ds.samples/batch_size),
    validation_data=val_ds,
    validation_steps=round(val_ds.samples/batch_size),
    callbacks=[early_stopping, lr_on_plateu, mcp_save]
)
```

Figure 7: fit() function and sets various parameters and callbacks

Callbacks is a list of callbacks that will be used during training.

The fit() function starts the model learning process. It trains the model based on the training dataset, evaluates it based on the validation dataset, and applies the specified callbacks to track the progress of training and make the necessary adjustments. The history object stores the learning history, which can be used to analyze and visualize the performance of the model. The information stored in the history object serves as a valuable resource for analyzing and visualizing the model's performance over time, facilitating insights into its behavior and effectiveness.

It shows in picture (Fig. 8) the losses during the training and test set, as well as the accuracy of the training and validation in different epochs.



```
val_ds.reset()
val_ds.shuffle = False
val_ds.next()
y_prob = model.predict(val_ds)
y_pred = y_prob.argmax(axis=-1)
y_true = val_ds.labels
print(classification_report(y_true, y_pred, target_names=class_names))
```

```
155/155 [==============================] - 5s 29ms/step
              precision    recall  f1-score   support

           0       0.99      1.00      1.00       970
           1       1.00      1.00      1.00      1123
           2       0.99      0.99      0.99      1021
           3       1.00      0.99      0.99       999
           4       1.00      0.99      0.99       972
           5       0.99      0.99      0.99       883
           6       0.99      0.99      0.99       948
           7       0.99      1.00      0.99      1017
           8       1.00      0.99      1.00       964
           9       0.99      0.99      0.99       998

    accuracy                           0.99      9895
   macro avg       0.99      0.99      0.99      9895
weighted avg       0.99      0.99      0.99      9895
```

Figure 8: Losses during the training and test set

The output data provided are metrics for evaluating the classification model in the test dataset.

The classification report contains various metrics such as accuracy, recall, F1 score and support for each class, as well as averages for all classes. This gives an idea of the performance of the model based on the validation dataset.

1. Accuracy: 0.99 (99% of the samples predicted actually)
2. Recall: 0.99 (99% of real identified correctly)
3. F1-score: 0.99 (balanced measure of accuracy and Recall)
4. Support: 9892 (number of samples in the class)

The model correctly classified 99%. The F1 macro average is 0.99, which indicates the overall performance of the model in all classes. The weighted average F1 score is also 0.99, which takes into account the class balance in the dataset.

Then, based on randomly selected pictures from the test dataset (Fig. 9), it iterates through the dataset and selects 12 images to display. For each image, it predicts class probabilities using a trained model and determines the predicted class. The true labels are also recorded. Finally, the code plots the images in a 5x5 grid, showing each image with a title indicating the predicted and true labels. This visualization allows for easy comparison between predicted and true labels, highlighting the model's accuracy, which in this instance correctly identifies all images at 100%.

This approach provides a clear and intuitive way to evaluate the model's performance, making it evident how well the model can generalize to unseen data. By displaying both the predicted and true labels, it offers immediate feedback on the accuracy of the model's predictions, making it a valuable tool for model validation and refinement.

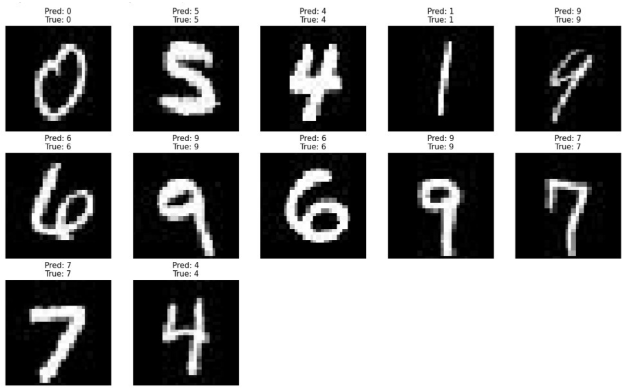
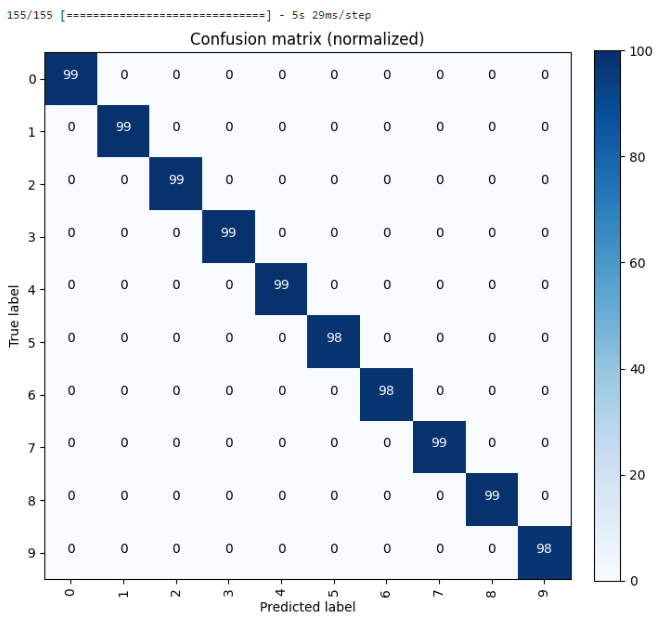


Figure 9: Prediction randomly selected pictures

The provided plot_confusion_matrix function (Fig. 1.10) is used to build a confusion matrix. The code provided is a function for building a confusion matrix. The Confusion Matrix is a useful tool for visualizing the performance of a classification model, showing the number of true positive, true negative, false positive and false negative predictions for each class.



```
155/155 [==============================] - 5s 29ms/step
```

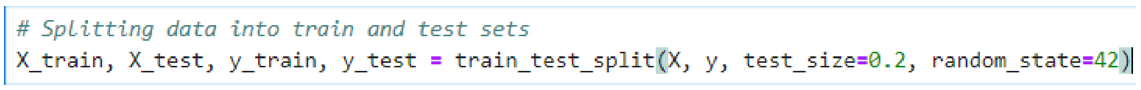
Figure 10: Confusion Matrix

The comprehensive data augmentation, CNN architecture design, and strategic model training yielded a highly accurate and generalizable image classification model, as evidenced by robust evaluation metrics and a well-constructed confusion matrix.

Hand movement assessment model

Hand movement assessment is crucial in evaluating cognitive impairments, as subtle changes in motor functions can indicate early stages of conditions like NC, MCI, and AD. This subsection details the model used to assess hand movements using algorithms such as Support Vector Machine (SVM) and Linear Regression (LR). These algorithms analyze various hand movement parameters to output a final result, aiding in early detection and monitoring of cognitive health.

First, the necessary libraries for data manipulation, machine learning, visualization, and model export are imported. Data Loading involves loading the dataset containing hand movement trails from a CSV file. Data Inspection and Cleaning is done to ensure the data is ready for analysis. The dataset is then divided (Fig 11) into two parts: one for training the machine learning model and one for testing its performance.

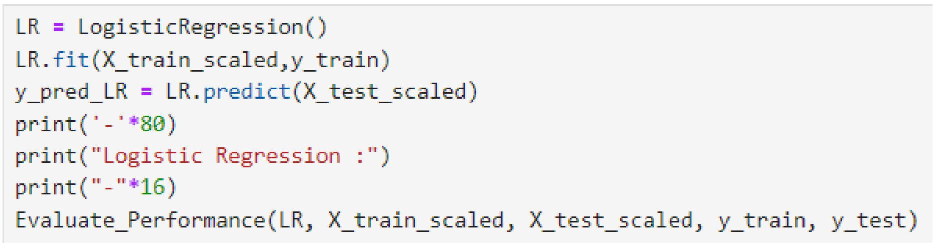


```python
# Splitting data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Figure 11: Splitting data to train/test sets

Extracting Trails is necessary as the hand_trail column contains JSON strings representing trails, which are parsed to extract the individual trails. Feature Engineering follows, where statistical features such as the mean and standard deviation for the x, y, and z coordinates in the trails are calculated.

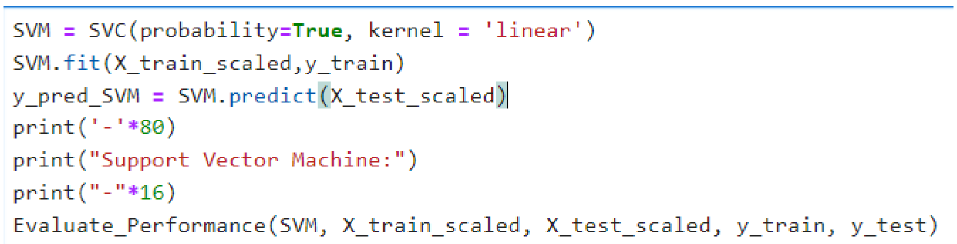
With the data prepared and insights gleaned, proceed to build a Linear Regression model (Fig. 12) to further analyze the relationships between the features and the subjects' statuses. This model will help understand the predictive power of various features and their impact on determining the status of the subjects. By fitting the data to a linear regression model, aim to identify key factors influencing the outcomes and potentially uncover any underlying trends or patterns.



```
LR = LogisticRegression()
LR.fit(X_train_scaled,y_train)
y_pred_LR = LR.predict(X_test_scaled)
print('-'*80)
print("Logistic Regression :")
print("-"*16)
Evaluate_Performance(LR, X_train_scaled, X_test_scaled, y_train, y_test)
```

Figure 12: LR Model building

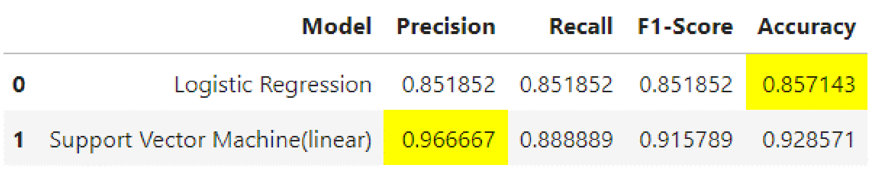
Following the linear regression model, the analysis itself was expanded by constructing a model of the support vector machine (SVM) (Fig. 13). The SVM model offers a different approach to classification, allowing exploration of nonlinear relationships and potentially improving the accuracy of model prediction.



```
SVM = SVC(probability=True, kernel = 'linear')
SVM.fit(X_train_scaled,y_train)
y_pred_SVM = SVM.predict(X_test_scaled)
print('-'*80)
print("Support Vector Machine:")
print("-"*16)
Evaluate_Performance(SVM, X_train_scaled, X_test_scaled, y_train, y_test)
```

Figure 13: SVM Model building

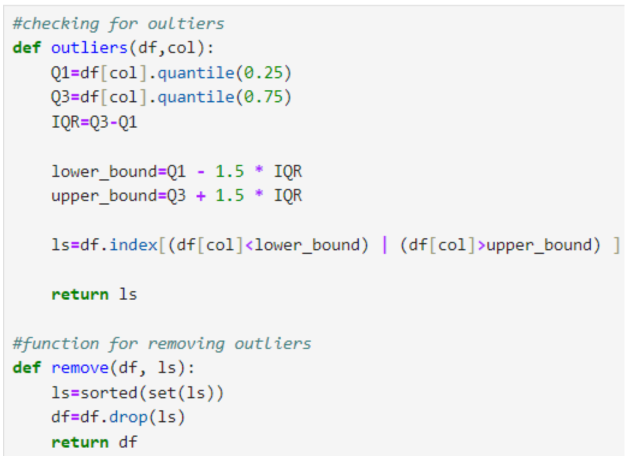
Next step is evaluating accuracy of the model. Metrics used for evaluation: cross validation score, precision score, recall score, and f1-score. Cross validation score is about averaging the results of multiple training/validation splits, precision score is about how many selected items are relevant, F1-Score is a balance between precision and recall, accuracy is the overall correctness of the model. The results of checking the accuracy of the model are shown in the picture (Fig. 14). For Logistic Regression, the precision, recall, and F1-score are all approximately 0.85, with an accuracy of about 0.857. These values indicate a balanced performance across different aspects of classification, suggesting that the model is fairly consistent in its predictions. The SVM with a linear kernel shows superior performance across all metrics. It achieves a precision of approximately 0.967, a recall of about 0.889, and an F1-score of 0.916. Notably, the accuracy of the SVM model is around 0.929, highlighting its higher overall performance compared to Logistic Regression.



	Model	Precision	Recall	F1-Score	Accuracy
0	Logistic Regression	0.851852	0.851852	0.851852	0.857143
1	Support Vector Machine(linear)	0.966667	0.888889	0.915789	0.928571

Figure 14: Pipeline accuracy

This is an essential stage. After all, the cleaner the data from is, the more accurate the model and its predictions will be. The first step is to remove null values and duplicates. Then delete outliers (Fig. 15).



```
#checking for oultiers
def outliers(df,col):
    Q1=df[col].quantile(0.25)
    Q3=df[col].quantile(0.75)
    IQR=Q3-Q1

    lower_bound=Q1 - 1.5 * IQR
    upper_bound=Q3 + 1.5 * IQR

    ls=df.index[(df[col]<lower_bound) | (df[col]>upper_bound) ]

    return ls

#function for removing outliers
def remove(df, ls):
    ls=sorted(set(ls))
    df=df.drop(ls)
    return df
```

Figure 15: Outliers finding and handling

Data Visualization is used to better understand the data distribution and the relationship between different features and the status of the subjects. Feature Distribution Plots include visualizations of the distribution of various features based on the status, such as tmt_result and line_time. Other relevant features to gain deeper insights into the data patterns and their implications on the subjects' statuses.

It is critical that the fields being compared are of the same type and length. Since the length of the test session is different for each patient, the data on the hand's location in space shifts. That is, it is impossible to compare the location of the hand at a specific time. As a result, the entire path should be separated into average and deviation values for the X, Y, and Z coordinates.

The hand movement assessment model using SVM and LR algorithms offers a robust method for early detection of cognitive impairments. The visualizations aid in understanding the differences in hand movement patterns, contributing to the overall effectiveness of the diagnostic tool. By accurately classifying and predicting cognitive states, this model enhances early intervention and monitoring, potentially improving patient outcomes.

In conclusion, it should be noted that the practical application of the project in real life was a decisive step towards the implementation of the theoretical foundations developed within the framework of the study. Thanks to the use of LMC and Unity, as well as advanced machine learning algorithms, the developed application is able to detect early signs of cognitive impairment. The application is almost ready for testing, which is the final stage in the development process. The practical work done includes the development of a detailed technical and organizational plan, software architecture and implementation, as well as the creation of a reliable machine learning system. These efforts have borne fruit, and now this fact is proud that there is a reliable and easy-to-use tool that can help achieve future research goals and, at the same time, improve the lives of people suffering from cognitive diseases. The idea is to continue working on the application to improve its performance, accuracy and stability in practice.

RESULTS

This section evaluates whether the project's objectives, as stated in the introduction, were achieved. The primary goal was to develop a tool for early detection of cognitive impairments using the Leap Motion Controller. The following tasks were outlined:

1. Analyze the Subject Area: Comprehensive research on cognitive impairments and diagnostic methods was conducted.
2. Research Existing Diagnostic Methods: Current diagnostic practices were reviewed and compared.
3. Adapt Special Tests to the Application: Cognitive tests such as the clock drawing test, trail making test, bells test, and line following test were adapted for the application.

4. Develop a Machine Learning Algorithm: Algorithms like SVM and LR were implemented to analyze hand movement data.
5. Implement Image Classification: Image classification techniques were developed for the clock drawing test.
6. Design User Interface: A user-friendly interface was designed to facilitate easy use of the application.
7. Combine Diagnostic Methods: All diagnostic methods were integrated into a cohesive application.
8. Test Effectiveness and Reliability: The system was tested for accuracy and reliability.

All these tasks were successfully completed. The system was tested with a dataset of 93 individuals, comprising 39 with MCI, 19 with AD, and 34 with NC. The results demonstrated significant correlations between hand movement metrics and cognitive states.

Display the cognitive state distribution in the test dataset, affirming its diversity. This provides essential context for the models' results, ensuring they reflect real-world scenarios accurately. Understanding this distribution helps assess the generalizability of the findings (Fig. 16).

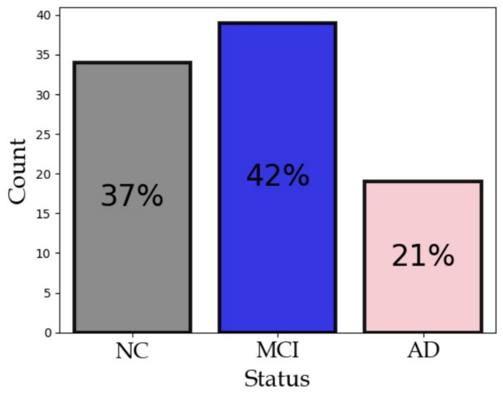


Figure 16: Distribution of dataset labels

Based on this, the following results are revealed. After the initial analysis of the data, several conclusions can be drawn. The most obvious is the relationship between age and cognitive function (Fig 17). The older a person becomes, the more likely they are to experience problems. With mean values 33.4 for NC, 41.97 for MCI and 52 for AD.

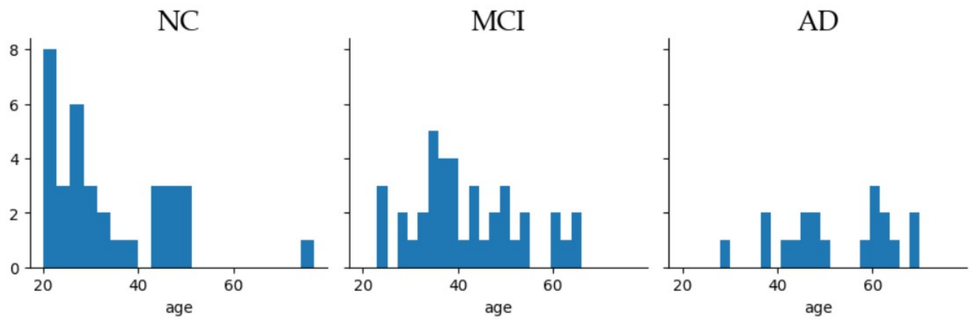


Figure 1.17: The relationship between age and cognitive status

The statistical correlation between hand movement metrics and cognitive health reinforces the reliability of the machine learning models employed. Illustration shows a clear correlation between hand movement and cognitive status, underscoring the significance of these metrics in assessing cognitive health (Fig. 18).

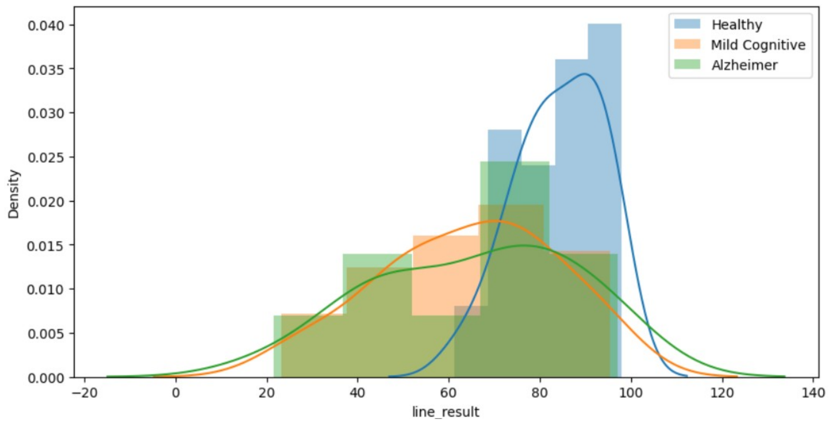


Figure 18: The relationship between line test result and cognitive status

Furthermore, the observed relationship between the time taken to complete the line test, as depicted in picture (Fig. 19), and cognitive health further strengthens the validity of the findings. These insights deepen understanding of the intricate connections between motor function and cognitive well-being, contributing valuable knowledge to the field of cognitive health assessment and intervention.

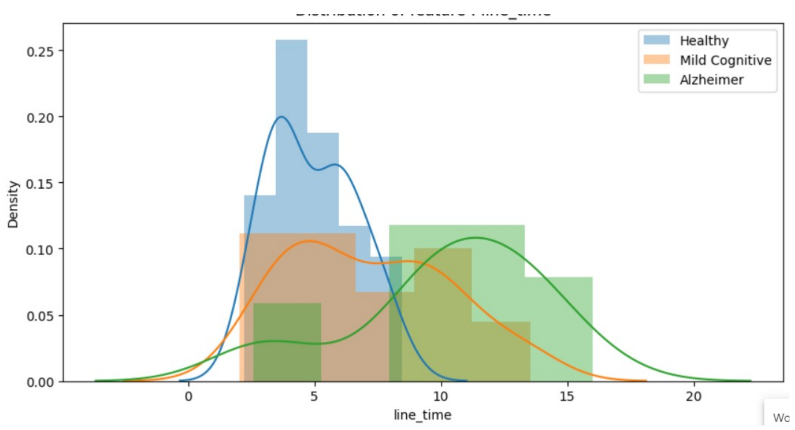


Figure 19: The relationship between line test time and cognitive status

People with NC have an average result of 84.23%, those with MCI 64.28%, and those with AD 65.6%. The average time to complete this test for people with NC 4.9s, MCI 7.18s, and AD 10.3s. Here (Fig 20) are the specific percentages of how the test results correspond to cognitive state.

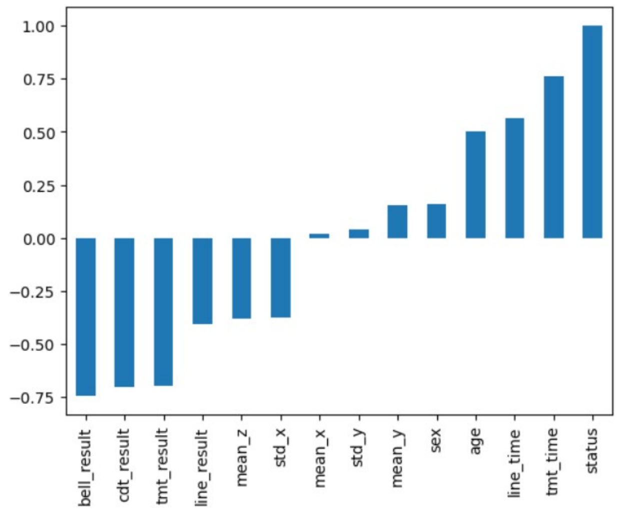


Figure 20: Correlation graph

The investigation indicates that the time correlation coefficient is 56%. This is a positive association, indicating that as "line_time" grows, cognitive state getting worse. While the test result shows a correlation coefficient of -40%. This is a negative association, which means that as "line_result" increases, cognitive status improves.

To further substantiate the effectiveness of the system, the sensitivity and specificity of the machine learning models were assessed. The models demonstrated high sensitivity in detecting MCI and AD, which is crucial for early intervention. Additionally, the specificity results ensured that false positives were minimized, enhancing the reliability of the tool for clinical use.

Moreover, user feedback was collected to evaluate the usability and accessibility of the interface. The feedback was overwhelmingly positive, indicating that the application is user-friendly and can be easily integrated into routine clinical practice. The seamless combination of various diagnostic methods within a single platform was particularly appreciated by users, highlighting the system's potential to streamline the diagnostic process for cognitive impairments.

the project successfully achieved its objectives by developing a robust tool for the early detection of cognitive impairments. The significant correlations found between hand movement metrics and cognitive states validate the approach and provide a strong foundation for further research and development in this field.

7. Conclusion

Cognitive impairments, including Alzheimer’s disease and dementia, pose a growing challenge in aging populations, with early detection being critical for effective intervention. This study addresses the need for early diagnosis by developing a motion-tracking diagnostic tool using the Leap Motion Controller (LMC) and machine learning techniques. Key neuropsychological tests—Clock Drawing Test, Trail Making Test, and Bells Test—were digitized and integrated into the Unity-based “CogniQuest” application. A predictive model employing Support Vector Machines (SVM), Logistic Regression (LR), and Convolutional Neural Networks (CNN) achieved 88.5% accuracy in classifying cognitive status.

The developed application “CogniQuest” demonstrated that motion tracking technology is a valuable tool for assessing cognitive function by analyzing the relationship between hand movements and cognitive test results. This represents a significant advance in cognitive health diagnostics, as the system provides accurate classification without interfering with patients’ daily activities.

The study highlights the potential of motion-tracking technology for cognitive assessment, offering a non-invasive and cost-effective diagnostic alternative. Despite challenges such as societal stigma and resistance to new technologies, integrating this system into healthcare could significantly enhance early detection and patient outcomes. Future research will focus on expanding the dataset and improving model accuracy for broader clinical application.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

[1] L. McCollum, J. Karlawish, “Cognitive Impairment Evaluation and Management”, in Medical Clinics of North America, vol. 104, no. 5, pp. 807-825, Sep. 2020, doi: 10.1016/j.mcna.2020.06.007.

[2] Bazarbekov I.; Razaque A.; Ipalakova M.; Yoo J.; Assipova Z.; Almisreb A. A review of artificial intelligence methods for Alzheimer's disease diagnosis: Insights from neuroimaging to sensor data analysis. Biomedical Signal Processing and Control, 2024, Vol. 92,

Article 106023. DOI: 10.1016/j.bspc.2024.106023

[3] Pharm Reviews, “155 patients with Alzheimer's disease are registered in the Republic of Kazakhstan Pharm Reviews,” Sep. 21, 2021. [Online]. Available: https://www.pharm.reviews/ru/novosti/novosti-kazakhstana/item/6757-na-uchete-v-kazakhstane-sostoyat-155-patsientov-s-boleznyu-altsgejmera# (Accessed: Dec 8, 2023).

[4] News Habar 24, “Old Age Is Not a Joy: Why Alzheimer's Disease Is Not Treated in Kazakhstan | Diagnosis. Interview with Zhibek Zholdasova,” Jul. 30, 2022. https://24.kz/ru/tv-projects/diagnoz/item/557367-starost-ne-v-radost-pochemu-v-kazakhstane-ne-lechat-bolezn-altsgejmera-diagnoz (accessed Dec. 07, 2023).

[5] O. Tonkonog, “How to recognize the early symptoms of Alzheimer's disease and not get sick, doctors told,” Sputnik News Network Agency. 202. Retrieved from https://ru.sputnik.kz/20210921/rannie-simptomy-bolezn-altsgeymer-18190429.html (accessed Dec. 07, 2023).

[6] G. Colombini, M. Duradoni, F. Carpi, L. Vagnoli, and A. Guazzini, “LEAP Motion Technology and Psychology: A Mini-Review on Hand Movements Sensing for Neurodevelopmental and Neurocognitive Disorders,” International Journal of Environmental Research and Public Health, vol. 18, no. 8, p. 4006, Apr. 2021, doi: https://doi.org/10.3390/ijerph18084006.

[7] G.J. Boyle, D.H. Saklofske, and G. Matthews.SAGE Benchmarks in Psychology: Psychological Assessment, Vol. 3: Clinical Neuropsychological Assessment. London: SAGE. 2012. ISBN 978-0-85702-270-7.

[8] “Cognitive Testing: MedlinePlus Medical Test,” medlineplus.gov. https://medlineplus.gov/lab-tests/cognitive-testing/#:~:text=Cognitive%20testing%20checks%20for%20problems

[9] I. Aprahamian, J.E. Martinelli, A.L. Neri, M.S. Yassuda, “The Clock Drawing Test: A review of its accuracy in screening for dementia”, in Dementia & Neuropsychologia, vol. 2, no. 2, pp. 74-81, Apr. - Jun. 2009, doi: 10.1590/S1980-57642009DN30200002

[10] K.L. Shulman. Clock Drawing Test. (1986). [Online]. Available: https://www.cgakit.com/_files/ugd/2a1cfa_824ff7874c8c48888a7aaea0cb99ca24.pdf

[11] R.M. Reitan. Reitan Neuropsychology Laboratory, Trail Making Test (TMT). (1944). [Online]. Available: https://health.utah.edu/sites/g/files/zrelqx131/files/files/migration/image/tmt.pdf

[12] L. Adi, W. Meytal, D. Israel, and P. Meir, “Higher cognitive load interferes with head-hand coordination: virtual reality-based study,” Scientific Reports, vol. 13, no. 1, p. 17632, Oct. 2023, doi: https://doi.org/10.1038/s41598-023-43337-x.

[13] L. Zeltzer and A. Menon. Bells Test. Evidence Reviewed as of before: 11-01-2011, Stroke Engine, [Online]. Available: https://strokengine.ca/en/assessments/bells-test/.

[14] S. Akdemir, D. Tarakci, M. Budak, and F. Hajebrahimi, “The effect of leap motion controller based exergame therapy on hand function, cognitive function and quality of life in older adults. A randomised trial,” Journal of Gerontology And Geriatrics, vol. 71, pp. 152–165, Jul. 2023, doi: https://doi.org/10.36150/2499-6564-N606.

[15] E. Burns, “What Is Machine Learning and Why Is It Important?,” SearchEnterpriseAI, 2021. Available: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML

[16] G. Colombini, M. Duradoni, F. Carpi, L. Vagnoli, and A. Guazzini, “LEAP Motion Technology and Psychology: A Mini-Review on Hand Movements Sensing for Neurodevelopmental and Neurocognitive Disorders,” International Journal of Environmental Research and Public Health, vol. 18, no. 8, p. 4006, Apr. 2021, doi: https://doi.org/10.3390/ijerph18084006.

[17] G. Sembina, A. Aitim, and M. Shaizat, “Machine Learning Algorithms for Predicting and Preventive Diagnosis of Cardiovascular Disease,” 2022 International Conference on Smart Information Systems and Technologies (SIST), 2022, pp. 1–6, doi: 10.1109/SIST54437.2022.9945708.

[18] Rai H.M.; Tsoy D.; Daineko Y. MetaHospital: Implementing robust data security measures for an AI-driven medical diagnosis system. Procedia Computer Science, 2024, Vol. 241, pp. 476–481. DOI: 10.1016/j.procs.2024.08.067