TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Software Science

Ivan Uvarov 164228 IAPB

ENHANCEMENT AND AUGMENTATION OF DRAWING TESTS FOR DEEP LEARNING BASED DIAGNOSTICS OF NEUROLOGICAL DISORDERS

Bachelor Thesis

Supervisor
Sven Nõmm
PhD
Co-supervisor
Elli Valla
MSc

Tallinn 2021

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

Ivan Uvarov 164228 IAPB

ANDMEKOGUMITE RIKASTAMINE NEUROLOOGILISTE HAIGUSTE TUVASTAMISEKS SÜGAVÕPPE MUDELITEGA

Bakalaureusetöö

Juhendaja
Sven Nõmm
PhD
Kaasjuhendaja
Elli Valla
MSc

Tallinn 2021

Author's declaration of originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author:	Ivan Uvarov	....
		(signature)
Date:	June 14th, 2021	

Abstract

Parkinsons' disease (PD) is a neurodegenerative disease often associated with symptoms such as tremor, bradykinesia, rigidity. Early detection of PD is difficult; however, in recent years, digital written tests using tablet computers have allowed us to collect kinematic and pressure features, which could also be analysed and help early detection of PD.

The primary goal of this thesis was to combine geometry based assessment with kinematic and pressure-based approaches. Enhancement and augmentation of the existing dataset of Archimedean spiral tests was performed, which was then used in CNN training. Another objective of this thesis was to find the most suitable enhancement combinations of parameters that would produce the highest accuracies in PD prediction using selected CNN architectures. In order to accomplish this goal a program was created, which would generate images from existing data and enhance these images with kinematic and pressure-based features, then augment the dataset and train the CNNs, with all training and evaluation data being logged.

The main results of this thesis are multiple enhancement combinations leading the deep neural network models to recognize PD with over 95% accuracy. Baseline architecture AlexNet returned 84.9% accuracy and after hyper-parameter tuning produced 95.4% accuracy. Also, pretrained model Xception had top 4 enhancement combinations all reach over 94% accuracy, the highest being 99.2% accuracy.

The results show that the enhancement of datasets, combining the geometric and feature-based approaches, can be a viable method in helping with the prediction of PD using CNNs.

The thesis is written in English and contains 25 pages of text, 4 chapters, 33 figures, 8 tables.

Annotatsioon

Parkinsoni tõbi (inglise keeles Parkinsons' disease - PD) on kesknärvisüsteemi haigus, mille peamisteks sümptomiteks on värinad, liigutuste aeglustumine ning lihaste jäikus. Parkinsoni tõve varajane tuvastamine on keeruline, kuid digitaalsete kirjutamis- ja joonistustestide kogumine tahvelarvutitega on andnud võimaluse koguda ka kinemaatilisi ja muid parameetreid, mida saab samuti analüüsida ning mis võivad olla abiks PD tuvastamisel.

Lõputöö põhieesmärgiks oli kombineerida geomeetrial baseeruv hindamine koos kinemaatiliste ja survepõhiste parameetrite lähenemisega. Selleks rikastatati ja suurendati (inglise keeles augment) olemasolev andmekogu Archimedeuse spiraali testidest, mida sai edasi kasutada konvolutsioonilise närvivõrgu treenimisel. Lisaeesmärk oli ka leida kõige sobivamad rikastamise parameetrite kombinatsioonid, mis annaks kõige kõrgema täpsuse ja tulemuse PD tuvastamisel väljavalitud konvolutsiooniliste närvivõrkude arhitektuuridega. Nende eesmärkide saavutamiseks loodi programm, mis genereeris algandmetest pildid ning rikastas need testide pildid kinemaatiliste ja survepõhiste parameetritega, seejärel andmekogu suurendati erinevate augmenteerimisvõtetega ning kasutati närvivõrgu treenimisel. Kõik treeningute andmed ja tulemused salvestati.

Lõputöö tulemused tõid esile mitmeid rikastamise kombinatsioone, mis andsid üle 95% täpsuse PD tuvastamisel. AlexNeti arhitektuur andis algselt 84.9% täpsuse ning peale hüper-parameetrite tuunimist andis kuni 95.4% täpsuse. Lisaks eeltreenitud (inglise keeles pre-trained) närvivõrgu mudel Xception andis 4 rikastamise kombinatsiooni, mille täpsused olid kõik üle 94% ning millest kõige kõrgem oli 99.2%.

Tulemused näitavad, et andmekogumite rikastamine, mis kombineerib nii geomeetrilist kui ka parameetrite põhist lähenemist, on abiks PD tuvastamisel.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 25 leheküljel, 4 peatükki, 33 joonist, 8 tabelit.

List of abbreviations and terms

PD	Parkinsons' disease
HC	Healthy control
CNN	Convolutional Neural Network
JSON	JavaScript Object Notation
IDE	Integrated development environment
PNG	Portable Graphics Format
RGB	Red Green Blue color model

Table of Contents

List of Figures vi

List of Tables viii

1 Introduction 1

1.1 Problem statement 2

2 Implementation 3

2.1 Overview 3

2.2 Image generation and enhancement 5

2.3 Augmentation 7

2.4 CNNs 8

3 Results 13

3.1 AlexNet 13

3.2 LeNet5 17

3.3 Xception 20

3.4 Comparison 21

4 Conclusion 24

Bibliography 27

Appendices 30

Appendices 30

List of Figures

1 *Full workflow* 5

2 *Sample drawing test data in a JSON file* 5

3 Original images of Archimedean spiral performed by a HC subject (a) and the PD patients (b, c) 6

4 *Example of the polygon drawing process* 6

5 Enhanced image of an Archimedean spiral with combination pressure_diff+velocity performed by a HC subject (a) and the PD patient (b) 7

6 Augmented images with OpenCV - original image (a), rotation (b), flipping (c), scaling (d, e) 8

7 Augmented images with Augmentor (shear and contrast are exaggerated for this figure) - original image (a), distortion (b), shear (c), contrast (d) . 8

8 *Splitting of data into train, validation and test sets.* 9

9 *AlexNet architecture [17]* 10

10 *LeNet5 architecture [18]* 11

11 *Xception architecture [19]* 11

12 *Train and validation accuracy of the best enhancement combination in AlexNet* 13

13 *Train and validation loss of the best enhancement combination using AlexNet* 14

14 *Validation accuracy of top performing combinations using AlexNet* 14

15 *Validation loss of top performing combinations using AlexNet* 14

16 *Train and validation accuracy of the best enhancement combination using AlexNet after tuning* 15

17 *Train and validation loss of the best enhancement combination using AlexNet after tuning* 15

18 *Validation accuracy of top performing enhancement combination using AlexNet after tuning* 16

19 *Validation loss of top performing enhancement combination using AlexNet after tuning* 16

20 *Training and validation accuracy of the best enhancement combination using LeNet5* 17

21 *Training and validation loss of the best enhancement combination using LeNet5* 17

22 *Validation accuracy of top performing combinations using LeNet5* 18

23 *Validation loss of top performing combinations using LeNet5* 18
24 *Training and validation accuracy of the best enhancement combination using modified LeNet5* 19
25 *Training and validation loss of the best enhancement combinations using modified LeNet5* 19
26 *Validation accuracy of top performing combinations using modified LeNet5* 19
27 *Validation loss of top performing combinations using modified LeNet5* . . 20
28 *Train and validation accuracy of the best enhancement combination for Xception* 20
29 *Train and validation loss of the best enhancement combination for Xception* 21
30 *Train and validation accuracy for altitude+jerk_y using Xception* 22
31 *Train and validation loss for altitude+jerk_y using Xception* 22
32 *Train and validation accuracy for pressure_diff+crackle using Xception* . 23
33 *Train and validation loss for pressure_diff+crackle using Xception* 23

List of Tables

1 Sample subset of vector features. 4

2 Top 10 enhancement combinations by test accuracy using AlexNet 13
3 Top 10 enhancement combinations by test accuracy for tuned AlexNet model 16
4 Top 10 enhancement combinations by test accuracy using LeNet5 17
5 Top 10 enhancement combinations by test accuracy using modified LeNet5 19
6 Top 10 enhancement combinations by test accuracy for Xception) 20
7 Top metrics for each tested architecture 22
8 Comparison with related works 23

1. Introduction

Parkinson's disease (PD) is a neurodegenerative disorder that mainly affects the motor system. It is the second most common neurodegenerative disorder with millions affected worldwide [1]. The most obvious symptoms of PD include tremor (involuntary muscle contractions), bradykinesia (slowness of movement) and rigidity. The cause of PD is unknown and currently there is no cure for PD [2]. However findings show that early treatment improves quality of life for the patients and caregivers [3].

Currently diagnosis of PD is based on the occurrence of the main symptoms such as tremor, bradykinesia and ruling out any other potential diseases with similar symptoms, which makes early detection of PD very difficult [2]. One of the ways to diagnose PD are handwritten tests such as writing sentences, drawing spirals, drawing Luria patterns [4]. Nowadays the data of such tests can be collected digitally with the use of tablet computers. This digital data gives access to more information behind the patients movements apart from the geometry of the drawing or text, which also helps in identifying PD.

In recent years machine-learning has become more prominent in diagnosis of PD. Multiple studies of written tests using machine-learning have shown promising results in identifying PD [5, 6, 7, 8]. Some studies have been based on kinematic parameters [5] and some on the geometry of tests [8].

The primary goal of this thesis was to combine geometry-based assessment with the novel approaches based on kinematic and pressure parameters registered during the drawing process and evaluate applicability of different CNN models to distinguish between the PD patients and HC subjects. Multiple parameters and enhancement combinations were found which produced good results.

The main outcomes of this thesis are enhancement combinations that provide up to 99.2% accuracy in PD prediction. Further comparison of different enhancement combinations and CNN architectures is also provided.

The thesis is divided into following parts. Introduction consists of the overview on the PD and a problem statement. Implementation section describes the implementation process to this thesis and technologies used. Results section provides the main outcomes of the CNNs with different enhancement combinations and also provides a comparison between the results of different CNNs used. Conclusion summarizes the thesis and its' results and explores possible improvements for future studies.

1.1 Problem statement

The present research aims to combine geometry-based assessment (used by a human practitioners) with the kinematic and pressure parameters-based approach (frequently used by machine learning shallow classifiers). This problem requires the following sub-problems to be solved.

1. Encode kinematic and pressure parameters into the drawn spiral image without affecting the geometry (shape of the drawing). Find the best subsets of the kinematic and pressure parameters to be encoded.
2. To avoid over-parameterization, find the suitable way for the data augmentation.
3. Evaluate the performance of a few most suitable deep CNN architectures for the case of the particular problem.

In order to encode parameters into the images a program was created, which would generate the image from the JSON data file and then enhance the image with two selected parameters. One of the parameters would indicate the width of the spiral and the other one would be used for the coloring of the spiral segments.

The data augmentation methods were manually tested and were then chosen for the augmentation flow with specified ranges.

Two famous CNN architectures, AlexNet and LeNet5, and a pre-trained model Xception, which is available in the Keras library, were chosen to be tested in this thesis. The results of all the trainings were logged and analysed using TensorBoard and then a comparison was done between the architectures.

2. Implementation

2.1 Overview

The data acquisition is described in detail in [9]. The data consisted of files describing HC (healthy control) and PD (patients with Parkinson's disease). The original set of data contained many different drawing tests per each person, but for this thesis only tests of Archimedean spirals were used, of which there were 31 for the HC group and 20 for the PD group, 51 samples in total.

All test data was in JSON format and contained recorded data of the drawing tests, which included metadata (session id, test type, anonymous identification number) and an array of data records containing dynamic features. The following dynamic features (time-sequences) were captured by the tablet: X-coordinate (mm); Y-coordinate (mm); timestamp (sec); pressure (arbitrary unit of force applied on the surface: [0,..., 6.0]); altitude (pen inclination, rad); azimuth (pen orientation, rad). Using these dynamic features, additional features could be extracted such as different derivatives of position, pressure and angles.

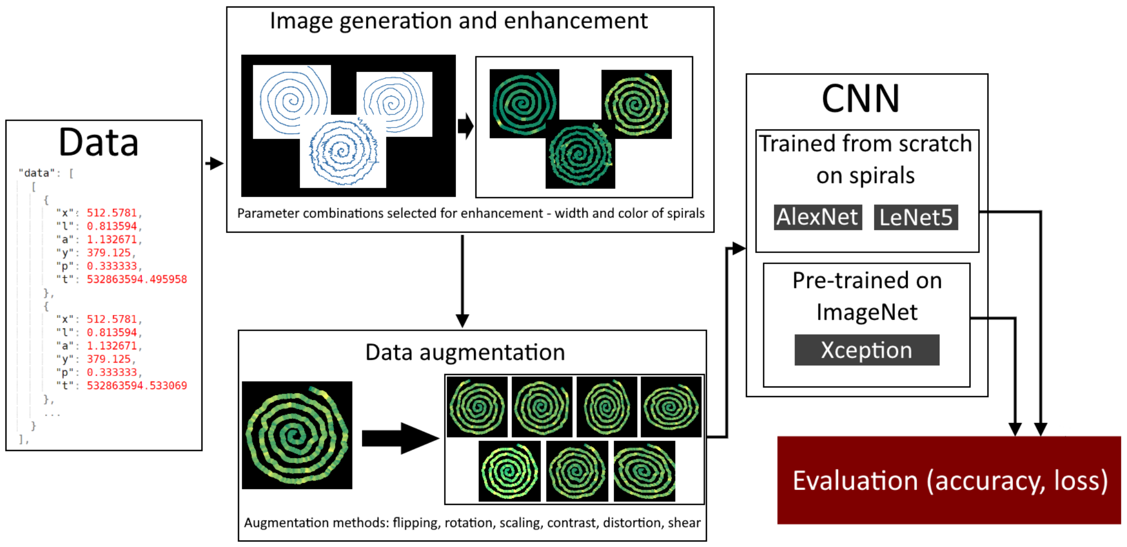
Feature set	Feature	Description
Spatial-temporal features	displacement	$d_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$
Kinematic features	velocity	Rate of change of displacement with respect to time. First time derivative of the displacement.
	acceleration	Rate of change of velocity with respect to time. Second time derivative of the displacement.
	jerk	Rate of change of acceleration with respect to time. Third time derivative of the displacement.
	snap	Rate of change of jerk with respect to time. Fourth time derivative of the displacement.
	crackle	Rate of change of snap with respect to time. Fifth time derivative of the displacement.
	pop	Rate of change of crackle with respect to time. Sixth time derivative of the displacement.
Pressure features	pressure_diff	Change of the pressure between points $[p_i, p_{i+1}]$
	yank	Change in pressure change between points $[p_i, p_{i+1}]$. First time derivative of the force applied on the surface.
	tug	Change in yank between points $[p_i, p_{i+1}]$. Second time derivative of the force applied on the surface.
	snatch	Change in tug between points $[p_i, p_{i+1}]$. Third time derivative of the force applied on the surface.
	shake	Change in snatch between points $[p_i, p_{i+1}]$. Fourth time derivative of the force applied on the surface.
Angular features	altitude_diff	Change in altitude between points $[p_i, p_{i+1}]$
	azimuth_diff	Change in the altitude acceleration between points $[p_i, p_{i+1}]$
	alphas_diff	Change in the alpha angle between points $[p_i, p_{i+1}]$
	yaw_diff	Change in the yaw angle between points $[p_i, p_{i+1}]$

Table 1. Sample subset of vector features.

The implementation was done using the Python programming language in the PyCharm IDE. The following are the additional libraries which were used extensively in the implementation:

1. NumPy and Pandas - open-source libraries for more effective computation in Python. These libraries were used for data storage, manipulation and overall utility [10, 11, 12].
2. MatPlotLib – a Python library used for visualization of data. This was the main library used for generating and enhancing images [13].
3. OpenCV – an open-source computer vision and machine learning software library. This was one of the libraries used in the augmentation of images, namely for rotation, scaling and flipping of images [14].
4. Augmentor – a Python library designed for augmentation and artificial generation of images for machine learning tasks. This library was also used in the image augmentation process [15].
5. TensorFlow – an open-source software library for machine learning. This library was mainly used for building and training the neural networks [16].

The full workflow consisted of removing any duplicate files in the dataset, then splitting the dataset into three separate subsets for the CNN training later on - training set, validation set and test set. This was done early in the workflow in order to avoid mixing images from the same patients in these subsets. After splitting the dataset, the JSON files were parsed and the images were generated and enhanced. Then all the images were augmented and we were left with a larger set of data. The augmented training set and validation set were then used to train and validate the CNN model. The training and validation results were logged after each epoch (iteration) of training. Finally after training, the model was evaluated using the test data and the results were logged into a separate file.

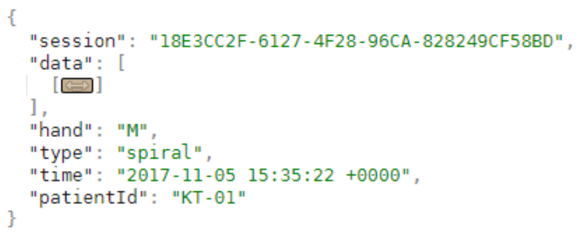

Data
Image generation and enhancement
Parameter combinations selected for enhancement - width and color of spirals
Data augmentation
Augmentation methods: flipping, rotation, scaling, contrast, distortion, shear
CNN
Trained from scratch on spirals
AlexNet
LeNet5
Pre-trained on ImageNet
Xception
Evaluation (accuracy, loss)

```
"data": [
  [
    {
      "x": 512.5781,
      "l": 0.813594,
      "a": 1.132671,
      "y": 379.125,
      "p": 0.333333,
      "t": 532863594.495958
    },
    {
      "x": 512.5781,
      "l": 0.813594,
      "a": 1.132671,
      "y": 379.125,
      "p": 0.333333,
      "t": 532863594.533069
    },
    ...
  }
],
```

Figure 1. *Full workflow*

2.2 Image generation and enhancement

The generation and enhancement of images was accomplished using Matplotlib library. The JSON data of a file was parsed into a Pandas dataframe and all additional features were then calculated. Two parameters were chosen for the enhancement – one for the width and the other for the color of the spiral. Both parameter arrays were normalized and a simple moving average calculation was applied to smoothen the transitions between drawn objects.



```
{
  "session": "18E3CC2F-6127-4F28-96CA-828249CF58BD",
  "data": [
    [⋯]
  ],
  "hand": "M",
  "type": "spiral",
  "time": "2017-11-05 15:35:22 +0000",
  "patientId": "KT-01"
}
```

Figure 2. *Sample drawing test data in a JSON file*

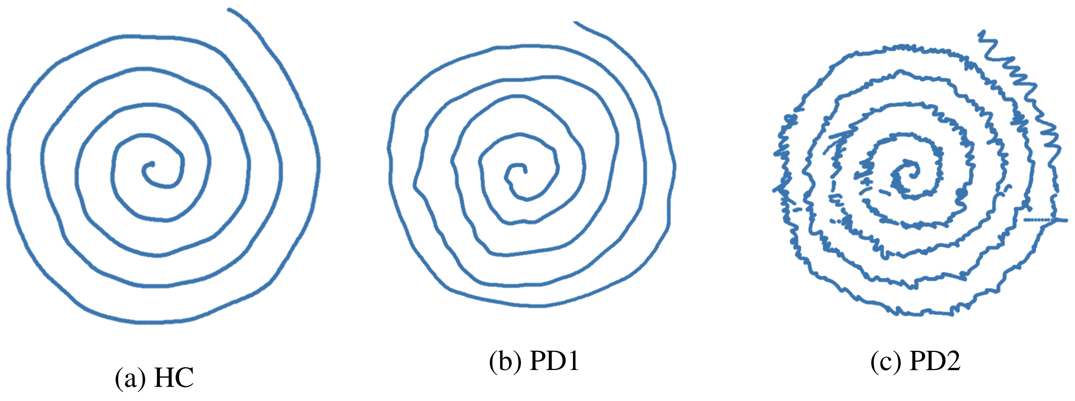


Figure 3. Original images of Archimedean spiral performed by a HC subject (a) and the PD patients (b, c)

The drawing of the spiral consisted of getting pairs of consecutive points and getting a line connecting these points. Additionally on the level on enhancing, finding perpendicular lines at both points and then finding vertices a width parameter away from the original points, finally connecting these points to create a polygon. These found vertices were then used in the next iteration with the next point to create the next polygon.

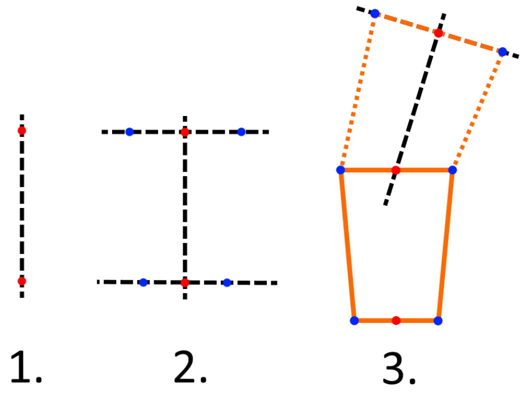


Figure 4. *Example of the polygon drawing process*

For the coloring of the polygons a Matplotlib colormap "summer" was chosen since the colors stood out from the black background and also could easily be converted to a grayscale image without losing much information. However any other sequential colormap from Matplotlib could also potentially be used as long as the colors of the colormap don't match the background of the image.

Once all the points were explored and all polygons were drawn, a solid black background was applied and the image was exported as a PNG file.

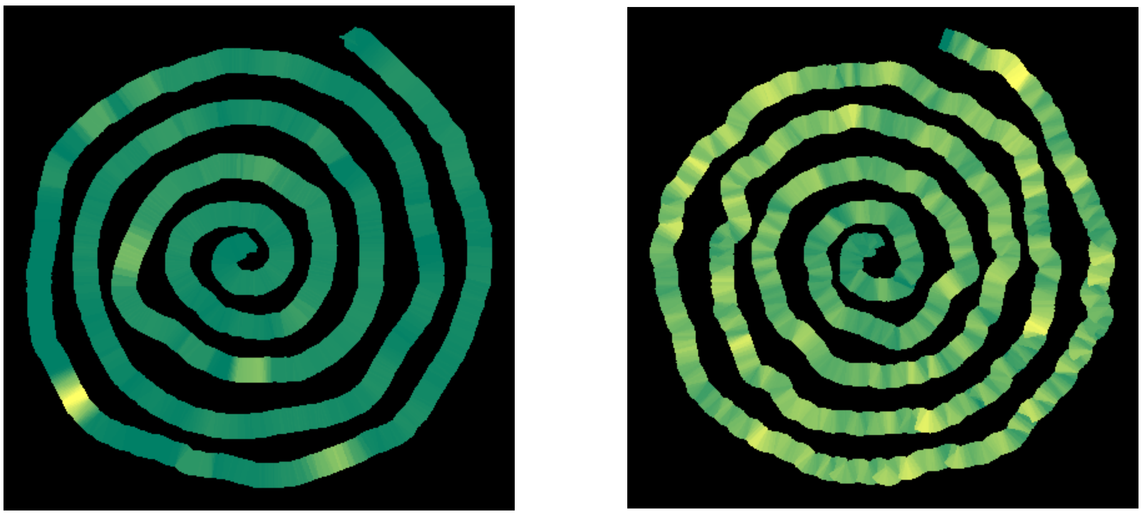

(a) HC (b) PD

Figure 5. Enhanced image of an Archimedean spiral with combination pressure_diff+velocity performed by a HC subject (a) and the PD patient (b)

2.3 Augmentation

With the images generated and enhanced, the next step was the augmentation of the dataset. In order to augment the images OpenCV and Augmentor libraries were used. The main augmentation techniques used were rotation, scaling and flipping using the OpenCV library [14] and additionally distortion, contrast and shear using the Augmentor library [15]. Since the shape of the spiral itself was also important, it was crucial to minimize the scale of changes to the overall geometry of the spiral.

Rotation was first done with 24 steps, each being 15 degrees, resulting in a range [0, 360) in the augmentation flow. Flipping was done horizontally, vertically and with both, which, including the image without flipping applied, produced 4 different images. However rotation was later changed to a range of [-30, 30] degrees with 13 steps, each step being 5 degrees, since some enhancement combinations were shown to depend on the orientation of the image.

Scaling consisted of compressing and stretching the image horizontally and vertically and then padding the resulting rectangular image with black borders in order to preserve the aspect ratio of the image. However, since this augmentation method made large changes to the overall geometry of the image, the scale of these changes was kept to a minimum. Scaling was done with 9 steps in range [0.8, 1.2], with the modifier showing the image width multiplier.

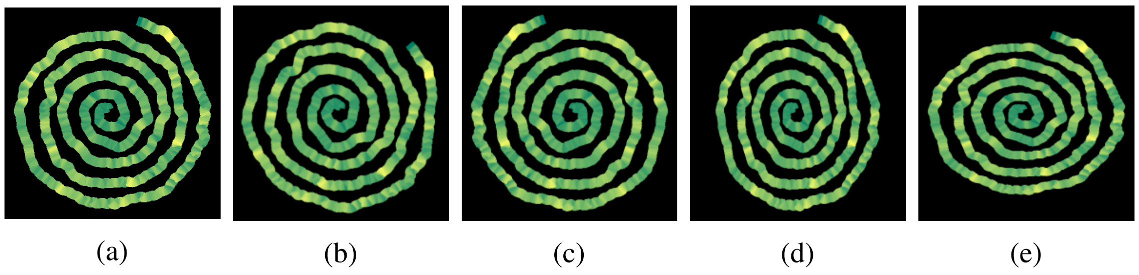


Figure 6. Augmented images with OpenCV - original image (a), rotation (b), flipping (c), scaling (d, e)

Augmentor library was also added later and has multiple different augmentation methods, however distortion was mainly used, with contrast and shear methods also used, but minimally. Distortion works by applying random elastic distortions to the image whilst preserving the size and aspect ratio of the image. The magnitude of these elastic distortions was kept to a minimum in order to preserve the overall geometry of the original image. Contrast was just changing the contrast of image. Shear consisted of sliding the top and bottom of the image in separate directions, transforming the square image into a parallelogram form and then coloring in the background.

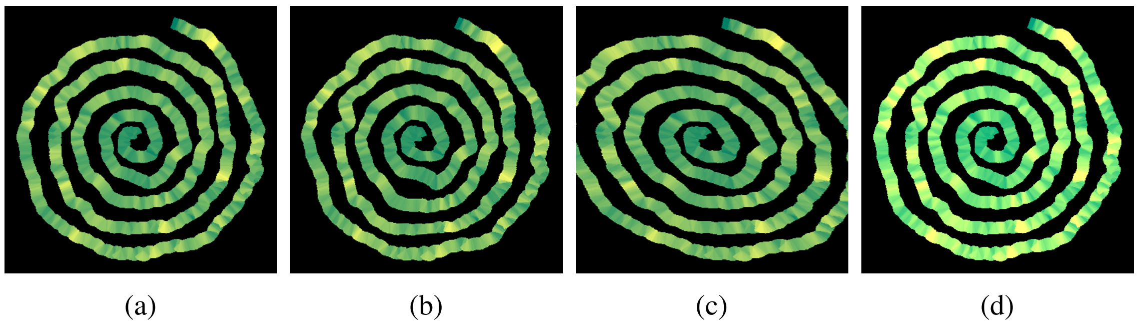


Figure 7. Augmented images with Augmentor (shear and contrast are exaggerated for this figure) - original image (a), distortion (b), shear (c), contrast (d)

Once the augmentation process was finished we were left with roughly 30000 augmented images to be used in CNN training.

2.4 CNNs

Convolutional neural network (CNN) are a class of deep neural network commonly used in image recognition and classification problems. CNNs, like other artificial neural networks, are inspired by biological neural networks of animals, using neurons and synapses. Major advantage of CNNs over other traditional algorithms is that CNNs use automated learning to optimize their filters, thus feature extraction is done by the CNN itself without prior knowledge and human intervention.

TensorFlow library was used to build and train the CNN models. TensorFlow is a powerful library created and used by Google for machine-learning related tasks [16]. Furthermore TensorBoard, a TensorFlow visualization toolkit, was used for the visualization of training results. The trainings initially ran for 10 epochs (full iteration over all samples), which was later changed to 20 epochs and also a few runs were tested with 50 epochs, however it was decided that all subsequent models would be trained for 20 epochs, since 10 epochs was too little to collect enough information about the trainings and 50 epochs didn't really produce any noticeable improvements compared to 20 epoch runs.

For training two famous CNN architectures and a single pre-trained model were chosen. Models were evaluated using the test subset of data, which CNNs had not seen during the training, and with accuracy and binary cross-entropy loss metrics.

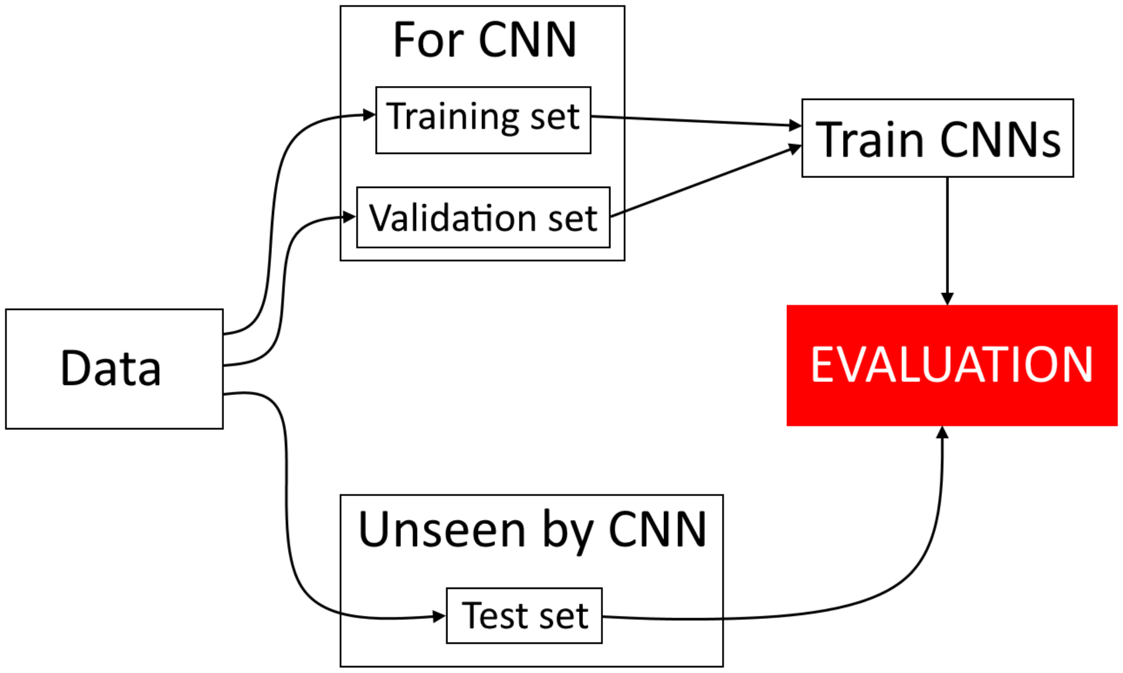


Figure 8. *Splitting of data into train, validation and test sets.*

Accuracy showed how often the predictions equaled labels using the formula

$$Acc = (TP + TN)/(TP + FP + FN + TN) \tag{2.1}$$

where TP - true positives, TN - true negatives, FP - false positives and FN - false negatives.

Binary cross-entropy, which is usually a go-to loss function for binary classification problems, showed loss between true labels and predicted labels and is calculated with the equation

$$L(\hat{y}, y) = -(y \log \hat{y} + (1-y) \log(1-\hat{y})) \tag{2.2}$$

where $\hat{y}$ is predicted value and y is true value.

The first CNN architecture used for the training was AlexNet and was a benchmark for other models later. AlexNet is a CNN architecture designed by Alex Krizhevsky et al. first published in the paper "ImageNet Classification with Deep Convolutional Neural Networks" in the year 2012. In 2012 AlexNet won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) and helped popularize CNNs. AlexNet takes the input images of size 227x227x3 (227x227 resolution with 3 RGB channels) and is made up of eight layers. The first five are convolutional layers, of which some are followed by max-pooling layers, and the last three are fully connected layers [17].

For this thesis the output layer (last layer in the architecture) was changed to 1 neuron, since we are dealing with a binary problem. Furthermore, the activation function for the last layer was changed from softmax to the sigmoid function, which tends to perform better for binary classification problems.

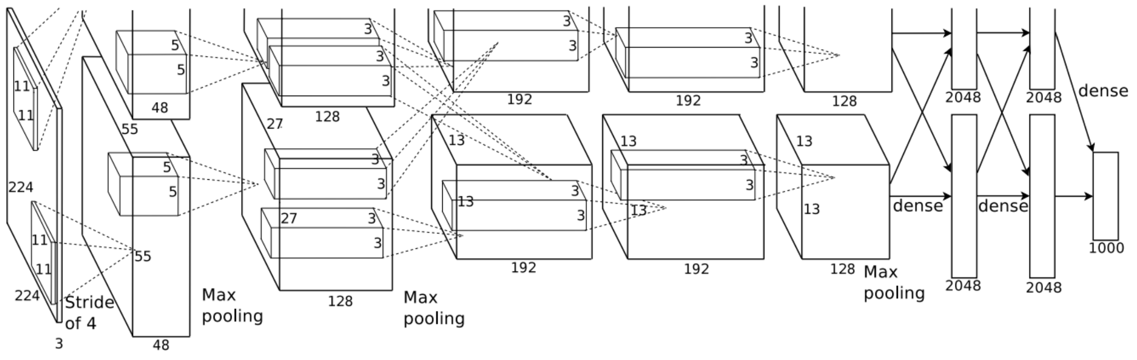


Figure 9. *AlexNet architecture [17]*

The second CNN architecture used was LeNet5, which inspired AlexNet. LeNet5 was introduced in the research paper “Gradient-Based Learning Applied To Document Recognition” in the year 1998 by Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. LeNet-5 takes the input of 32x32x1 (32x32 resolution with 1 RGB channel) and is made up of 7 layers, which consist of 3 convolutional layers, 2 subsampling layers and 2 fully connected layers [18]. Similarly to AlexNet, the output layer was changed to 1 neuron with sigmoid activation function.

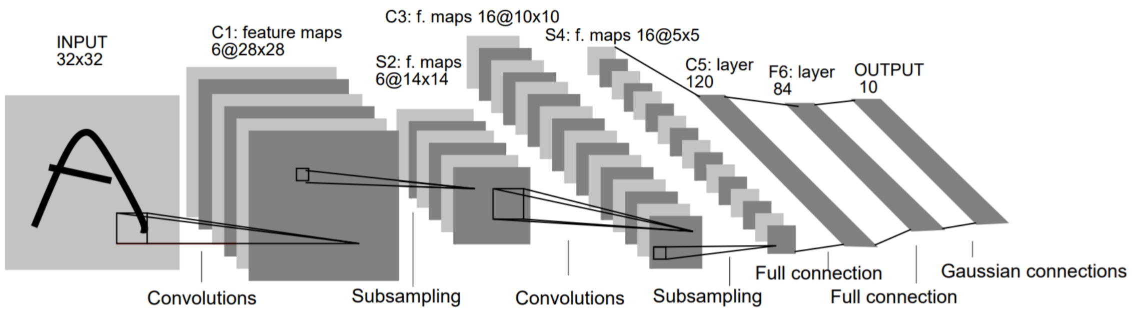


Figure 10. *LeNet5 architecture [18]*

Additionally, the pre-trained model Xception in the Keras library was also used to train on the dataset. Xception is a CNN architecture inspired by the Inception CNN architecture and was created by the creator of Keras [19]. This Xception model was pre-trained with data from ImageNet, which is a very large image database consisting of millions of annotated images, designed to be used in visual object recognition software research [20]. Other pre-trained models available in the Keras library such as VGG19, ResNet50 and Inception v3 were also considered initially, however Xception was chosen since it had one of the highest performances on the ImageNet validation dataset.

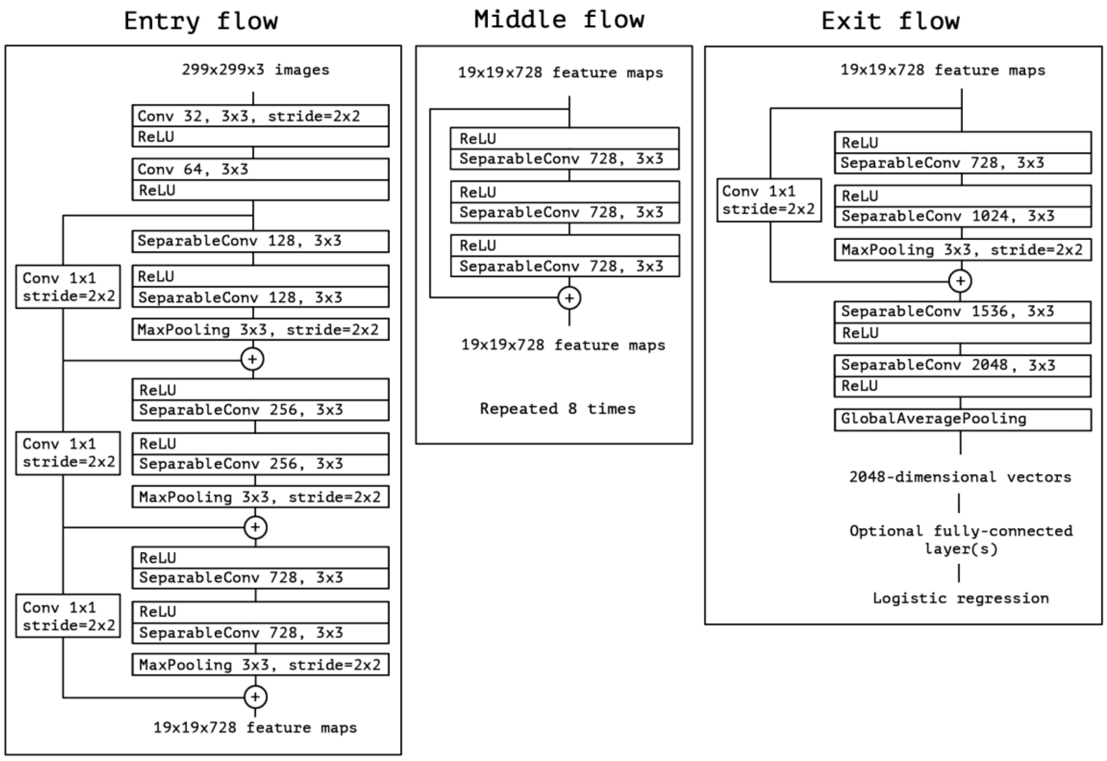


Figure 11. *Xception architecture [19]*

With Xception transfer learning was utilized, which is the reuse of a pre-trained model on a new problem. This way the model exploits the knowledge gained from a previous tasks to improve generalization when dealing with another task. For Xception model, the images were upscaled to 299x299 resolution since the input layer required a 299x299x3 input. The output layer was also changed to 1 neuron with a sigmoid activation function.

A total of 252 different enhancement parameter combinations were tested with AlexNet and LeNet5, but only select few combinations were tested using the Xception architecture. The combinations for the Xception model were selected from the top performing results obtained from AlexNet and LeNet5.

3. Results

This chapter gives an overview of the results obtained from CNNs using different enhancement combinations. A comparison of the results is done at the end of this chapter.

3.1 AlexNet

Inital runs with AlexNet showed good results, with test accuracy over 80% for multiple different enhancement combinations.

Combination	Accuracy	Loss
altitude+velocity_y	84.92%	1.63935864
shake+velocity	81.96%	1.17658496
pressure_diff+velocity	81.91%	2.05381775
pressure_diff+crackle	81.59%	2.23782539
altitude+velocity_x	80.34%	0.88364506
altitude+pop	80.10%	2.05658054
alphas+acceleration_x	79.62%	0.68236512
pressure_diff+snap	78.80%	1.12813747
altitude+crackle	78.76%	0.80387920
altitude+acceleration_y	78.35%	1.03007495

Table 2. Top 10 enhancement combinations by test accuracy using AlexNet

However, a closer look at the behaviour of validation metrics during training showed that the model had large fluctuations in its' validation accuracy and loss. Furthermore there was a massive gap between training and validation metrics.

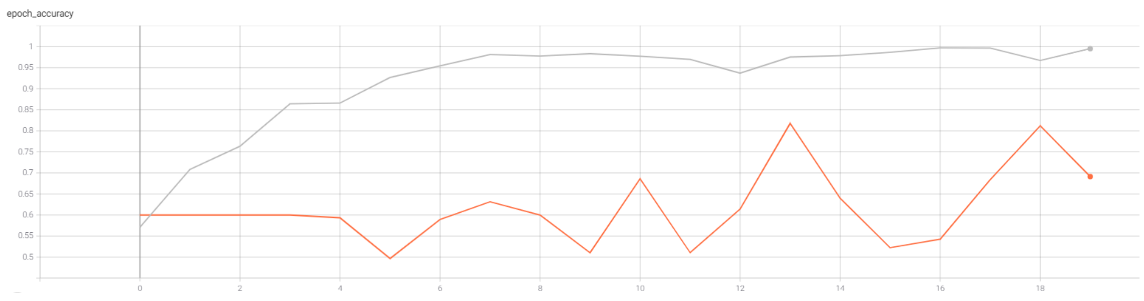


Figure 12. *Train and validation accuracy of the best enhancement combination in AlexNet*

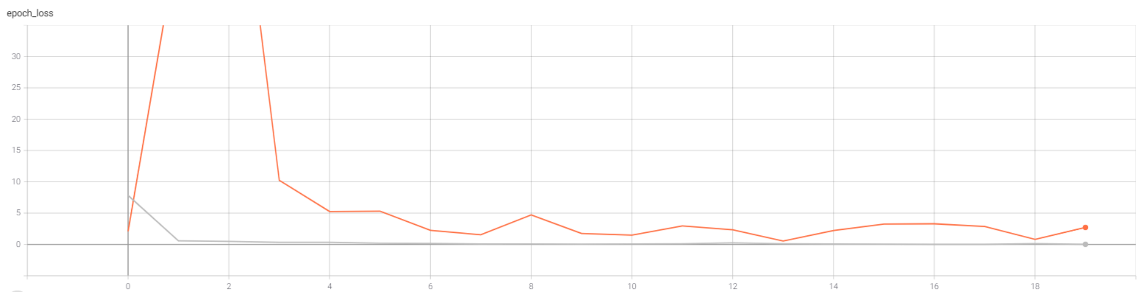


Figure 13. *Train and validation loss of the best enhancement combination using AlexNet*

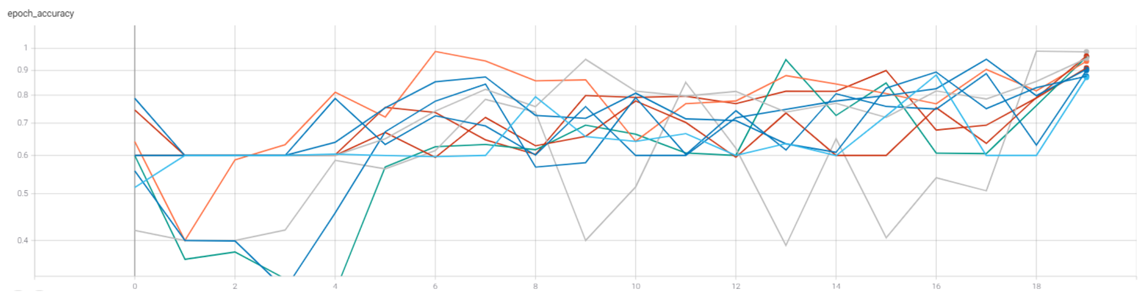


Figure 14. *Validation accuracy of top performing combinations using AlexNet*

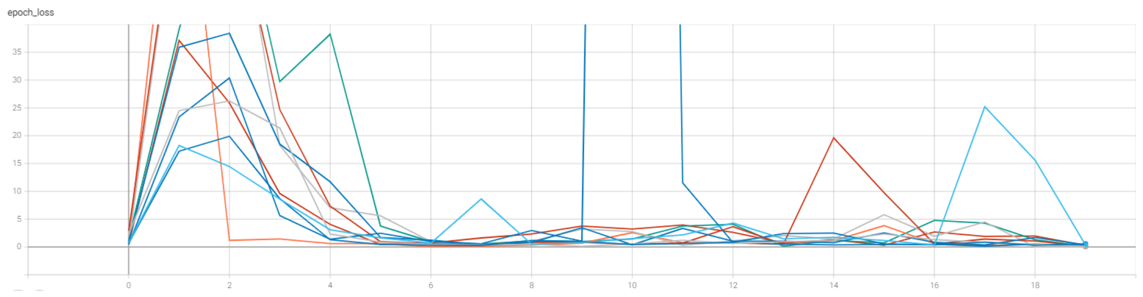


Figure 15. *Validation loss of top performing combinations using AlexNet*

This showed that the model was overfitting meaning the model was learning (memorizing) the training data too well and could not generalize to new data, thus negatively impacting the models' performance. This could be caused by the fact that augmented images from the same source image are very similar and differ only marginally. In order to reduce overfitting, adjustments were made to the augmentation process, the CNN model and the training process.

Firstly augmentation was checked. Rotation and scaling were adjusted, since these augmentation methods produced the biggest changes to the image. Rotation was changed from a full 360 degrees, to a range of [-30, 30] degrees and scaling multipliers were reduced. These changes were made since some enhancement combinations depended on the orientation of the images and the initial shape of the image. Additionally Augmentor library, with its' multiple augmentation methods, was added to the workflow at this point. This added another layer of augmentations to the dataset.

Learning rate of the model was also tuned. The training algorithms are sensitive to the change of the learning rate, so usually in practice it is the first hyper-parameter to be tuned. However, it is very hard to find the optimal learning rate on the first try. If the learning rate is too low, then the model will take a long time to converge or might get stuck in a local minimum. Conversely if the learning rate is too high, then the model might jump over the best configurations or even diverge. To combat this issue a learning rate scheduler was added to the model, which would start at a higher learning rate at the beginning of the training and then would progressively decrease the learning rate after each epoch.

Furthermore, L2 regularization was added to the model layers. Regularization is a process of introducing additional information in order to prevent overfitting.

These changes helped reduce the fluctuations in the validation accuracy and decrease the gap between training and validation loss.

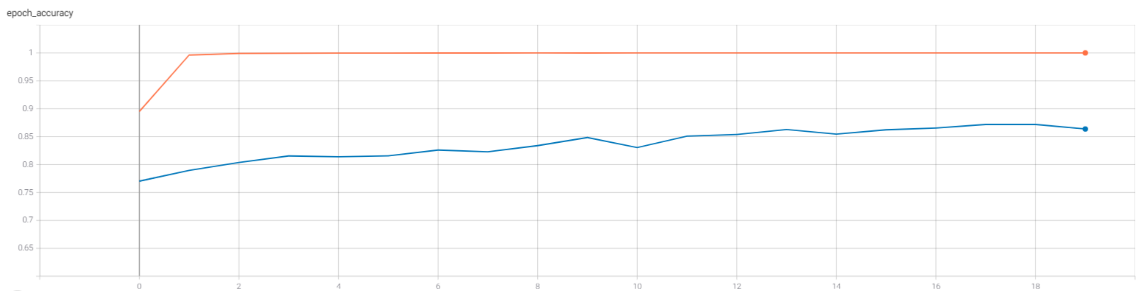


Figure 16. *Train and validation accuracy of the best enhancement combination using AlexNet after tuning*

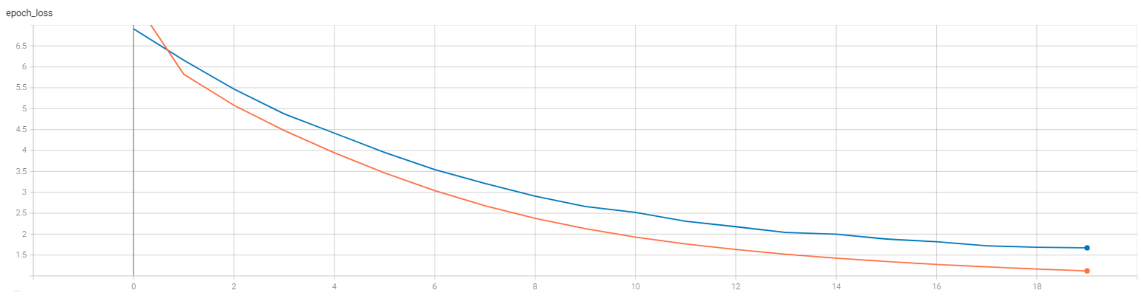


Figure 17. *Train and validation loss of the best enhancement combination using AlexNet after tuning*

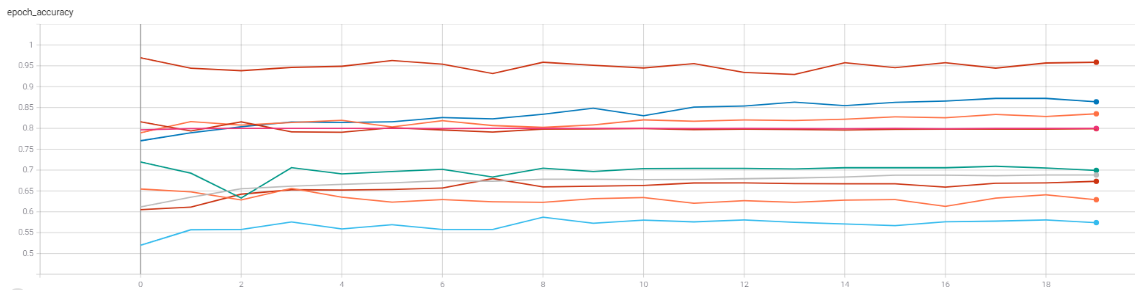


Figure 18. *Validation accuracy of top performing enhancement combination using AlexNet after tuning*

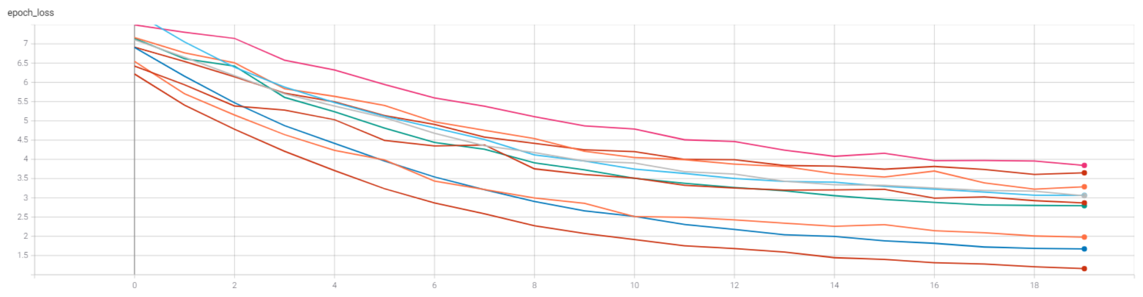


Figure 19. *Validation loss of top performing enhancement combination using AlexNet after tuning*

As can be observed, the fluctuations in overall validation accuracy and loss decreased. The model was still overfitting, however the results were nevertheless much better compared to prior results.

Combination	Accuracy	Loss
yaw_velocity+jerk	95.44%	1.21942651
yaw_velocity+jerk_y	92.92%	1.42715907
yaw_velocity+jerk_x	91.45%	1.74412179
azimuth_diff+jerk_y	86.32%	1.76336122
pressure_diff+jerk_y	84.54%	2.19594479
pressure_diff+velocity	84.17%	2.32450724
pressure_diff+pop	83.71%	1.75523186
snatch+jerk	83.43%	1.84873104
altitude+velocity_y	82.85%	2.41329455
alphas+acceleration_y	82.45%	2.29856396

Table 3. Top 10 enhancement combinations by test accuracy for tuned AlexNet model

After tuning all top 10 enhancement combinations managed to reach over 80% accuracy, with 3 combinations going over 90%. Using this tuned AlexNet model the combination of features yaw_velocity for width and jerk for color returned the best results with test accuracy reaching 95.44% and test loss 1.2194.

3.2 LeNet5

LeNet5 performed worse than AlexNet, producing overall lower test accuracy, however test loss was lower than AlexNet. Nonetheless, the top performing enhancement combination for LeNet5 managed to reach a higher accuracy than the AlexNet model prior to tuning.

Combination	Accuracy	Loss
pressure_diff+jerk_y	85.55%	0.40114945
pressure_diff+crackle	75.28%	0.49678519
azimuth_jerk+jerk	75.21%	0.69773960
pressure_diff+pop	74.55%	0.54215372
pressure_diff+jerk_x	73.54%	0.50928342
azimuth_acceleration+snap	73.01%	0.88120717
pressure_diff+snap	72.28%	0.54569393
azimuth_diff+jerk_y	72.28%	0.56830901
yaw_velocity+jerk_y	71.61%	0.50966978
azimuth_diff+snap	71.56%	0.65537918

Table 4. Top 10 enhancement combinations by test accuracy using LeNet5

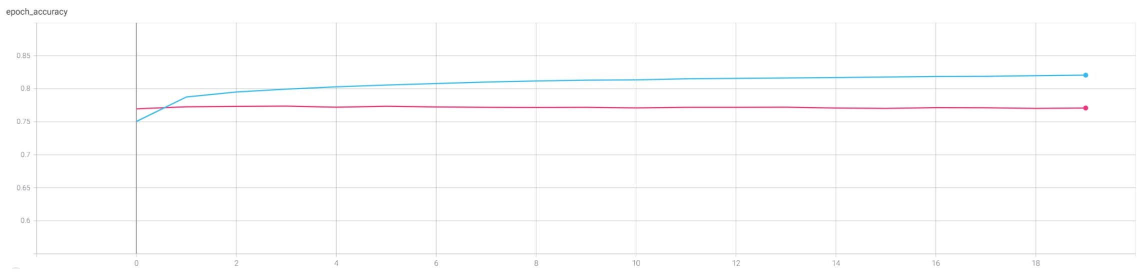


Figure 20. *Training and validation accuracy of the best enhancement combination using LeNet5*

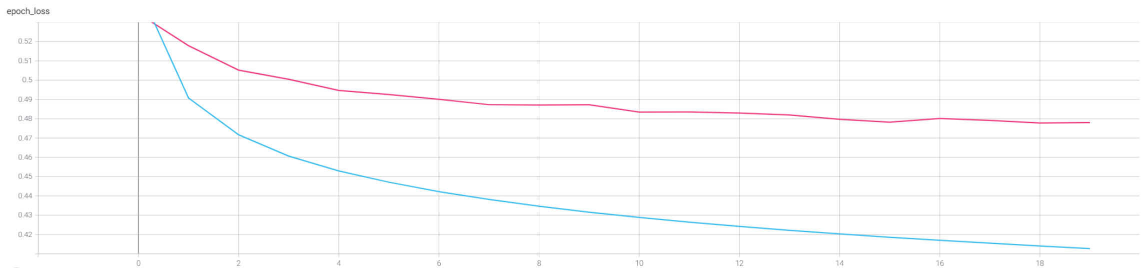


Figure 21. *Training and validation loss of the best enhancement combination using LeNet5*

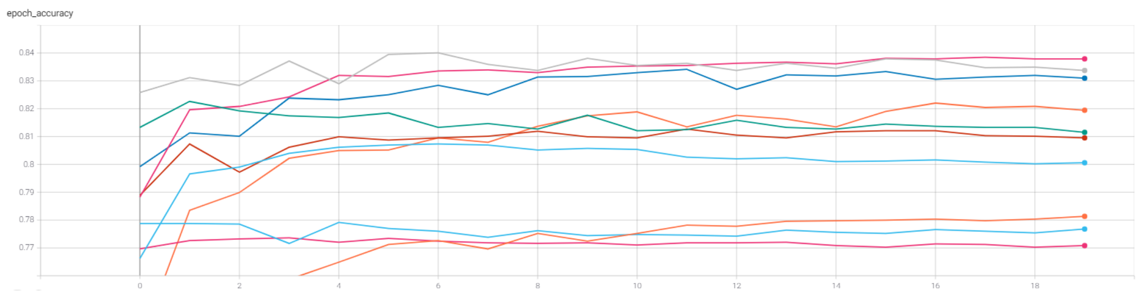


Figure 22. *Validation accuracy of top performing combinations using LeNet5*

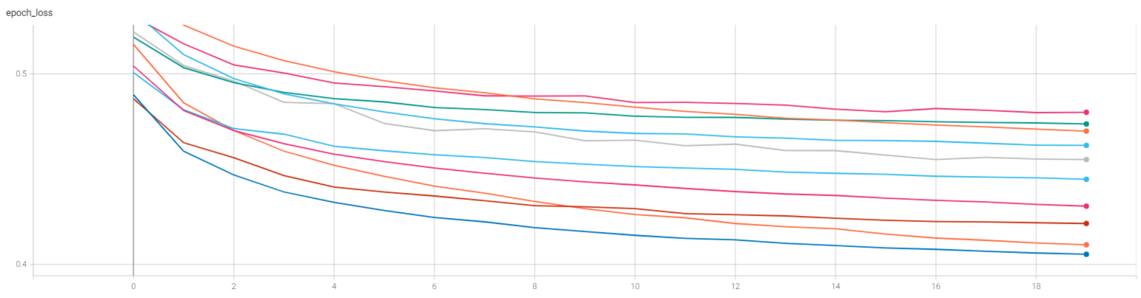


Figure 23. *Validation loss of top performing combinations using LeNet5*

The enhancement combination of pressure_diff for width and jerk_y for color produced the best results with 85.55% accuracy and 0.4011 loss and was the only combination to reach over 80% accuracy.

Further training was also conducted with a slightly altered LeNet5 architecture. The input layer of this model was modified to receive a larger input of 227x227x1, same as AlexNet except for the color channel, which stayed grayscale.

This modified LeNet5 model returned slightly higher average test accuracy, however the best enhancement combination with the modified LeNet5 had a slightly lower accuracy than the previous LeNet5 model.

Combination	Accuracy	Loss
altitude+velocity	0.84639549	1.01893830
altitude+acceleration_x	0.80572093	0.94429910
pressure_diff+acceleration_y	0.79993385	1.22108769
altitude+jerk	0.79133600	0.76885098
altitude+jerk_y	0.79001325	2.67037845
altitude+velocity_x	0.78240740	1.96160078
shake+acceleration_y	0.78009260	0.68461698
altitude+crackle	0.77959657	0.97345722
altitude+snap	0.77860451	1.20316017
shake+acceleration	0.77215606	1.81520116

Table 5. Top 10 enhancement combinations by test accuracy using modified LeNet5

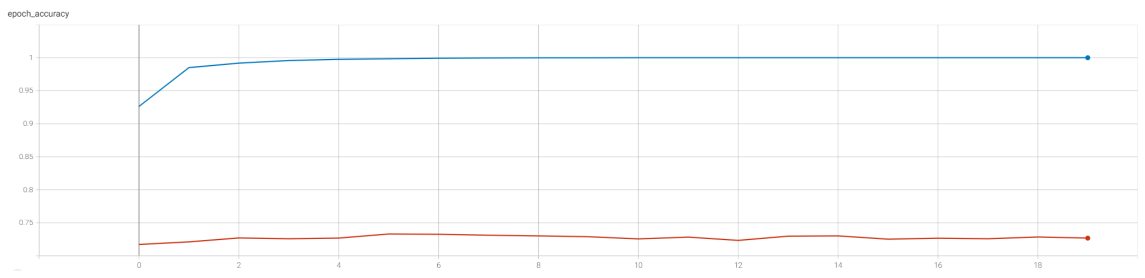


Figure 24. *Training and validation accuracy of the best enhancement combination using modified LeNet5*

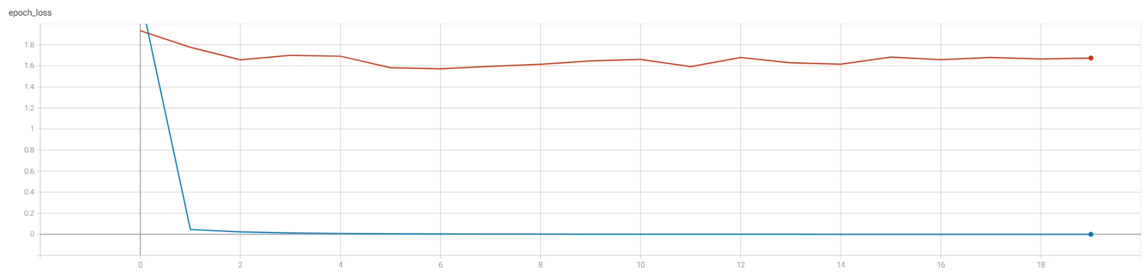


Figure 25. *Training and validation loss of the best enhancement combinations using modified LeNet5*

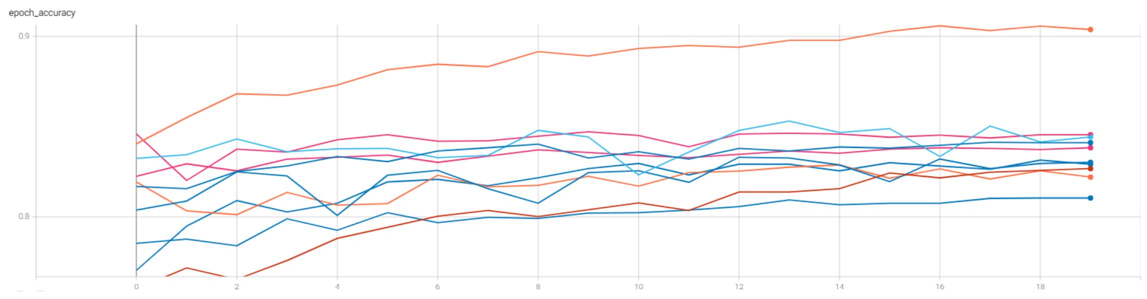


Figure 26. *Validation accuracy of top performing combinations using modified LeNet5*

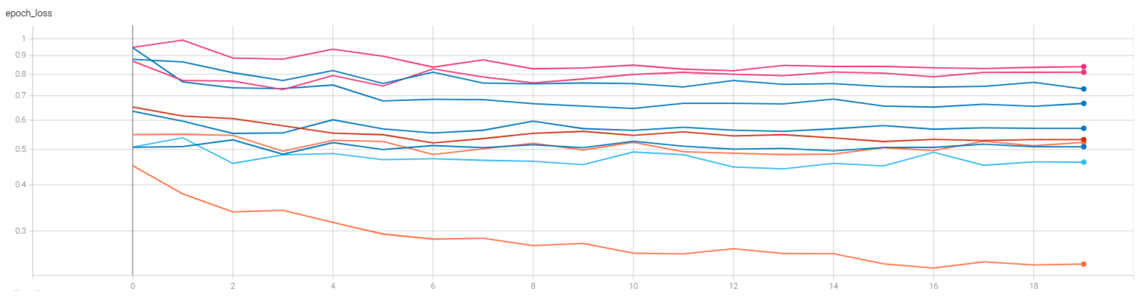


Figure 27. *Validation loss of top performing combinations using modified LeNet5*

Using this modified LeNet5 model the best enhancement combination was altitude for width and velocity for color, reaching 84.6% test accuracy and 1.089 test loss.

3.3 Xception

Xception model produced excellent results, boasting both high validation accuracy and test accuracy for a few parameter combinations.

Combination	Accuracy	Loss
yaw_velocity+jerk_x	99.20%	0.61184025
snatch+jerk	99.09%	0.61153144
yaw_velocity+jerk_y	97.47%	0.62166375
pressure_diff+velocity	97.47%	0.61460805
altitude+jerk_y	95.40%	0.61937094
yaw_velocity+jerk	94.89%	0.62052906
alphas+acceleration_x	94.37%	0.61867803
pressure_diff+pop	90.14%	0.62614924
altitude+pop	87.45%	0.63023144

Table 6. Top 10 enhancement combinations by test accuracy for Xception)

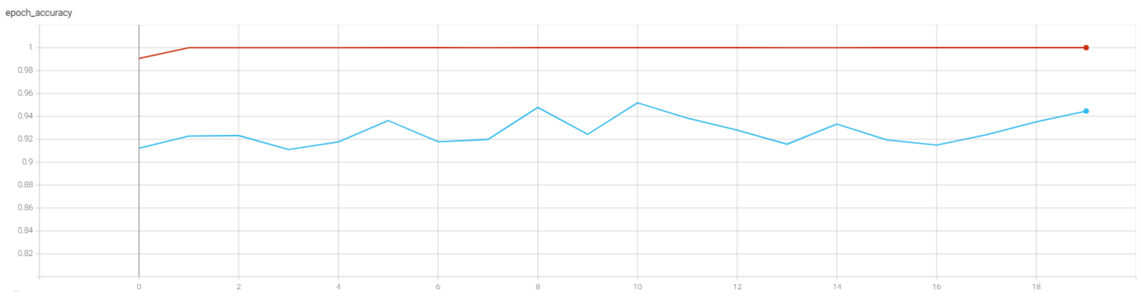


Figure 28. *Train and validation accuracy of the best enhancement combination for Xception*

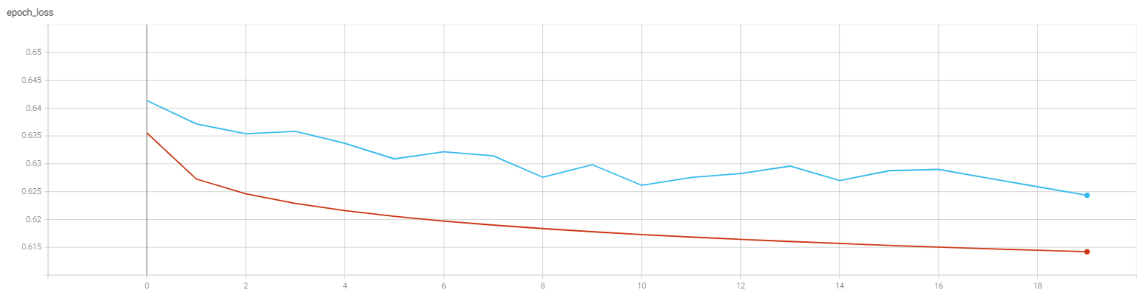


Figure 29. *Train and validation loss of the best enhancement combination for Xception*

However it can be observed, that there is still a gap between train accuracy and validation accuracy, with validation accuracy being lower. Although this gap was not very large, it still indicated that the model was slightly overfitting on the top performing enhancement combination.

Using Xception model and utilizing transfer learning the best result was an enhancement combination of yaw_velocity for width and jerk_x for color, which returned a 99.20% accuracy and 0.6118 loss.

3.4 Comparison

The first implementation of AlexNet gave decent results, the best result being the enhancement combination of altitude and velocity_y giving 84.9% test accuracy. However validation accuracy was fluctuating and the model was overfitting a lot.

After tuning, AlexNet returned better results than before, with top performing combination of yaw_velocity and jerk reaching 95.4% test accuracy. The model was still overfitting, but not as much as before the tuning.

The first LeNet5 model with 32x32x1 input returned by far the worst results of all the CNNs trained, with only one combination pressure_diff and jerk_y getting over 80% test accuracy, reaching 85.5% test accuracy.

The second LeNet5 model with larger input performed slightly better than the first LeNet5 model and was on par with the first AlexNet model, however still performed worse than the tuned AlexNet model. This LeNet5 model had a higher average test accuracy among the top performing enhancement combinations compared to the first LeNet5 model. However, the best combination using altitude and velocity had a lower test accuracy (84.6%) than the best performing combination in the first LeNet5 model (85.5%).

Xception returned by far the best results of all the CNNs. The top performing combination yaw_velocity and jerk_x reached 99.2% test accuracy. The average test accuracy of top combinations for Xception was also the highest among the CNNs trained.

Overall, both LeNet5 models and AlexNet model prior to tuning produced similar result with highest results reaching around 85% accuracy. Xception and tuned AlexNet model managed to get over 95% accuracy, with highest reaching 99.2% and 95.4% respectively.

Architecture	Highest accuracy	Best enhancement combination
Xception	99.2%	yaw_velocity+jerk_x
Tuned AlexNet	95.4%	yaw_velocity+jerk
LeNet5	85.5%	pressure_diff+jerk_y
AlexNet	84.9%	altitude+velocity_y
Modified LeNet5	84.6%	altitude+velocity

Table 7. Top metrics for each tested architecture

Some of the enhancement parameters appeared in multiple top performing enhancement combinations across all CNNs. For the width of the spiral these parameters included altitude, pressure_diff and yaw_velocity, and for the color of the spiral jerk_y, crackle and jerk.

It is worth mentioning that combinations altitude+jerk_y and pressure_diff+crackle were fitting very well with the Xception model and also produced decent results.

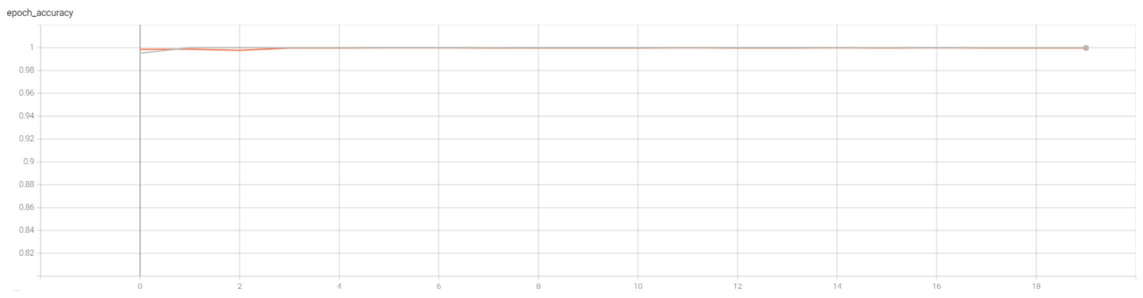


Figure 30. *Train and validation accuracy for altitude+jerk_y using Xception*

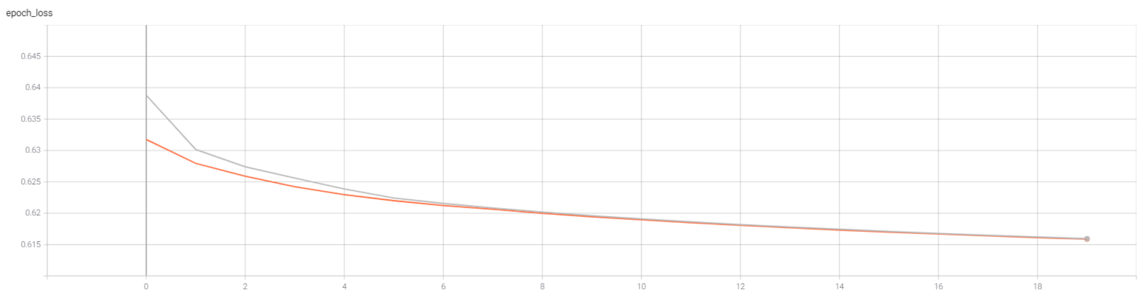


Figure 31. *Train and validation loss for altitude+jerk_y using Xception*

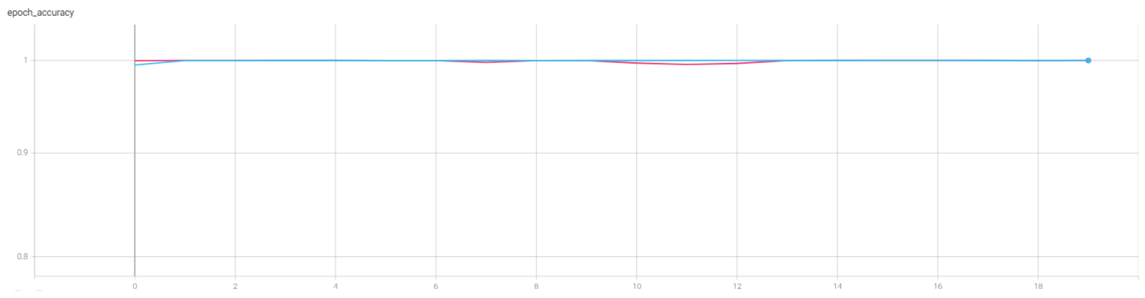


Figure 32. *Train and validation accuracy for pressure_diff+crackle using Xception*

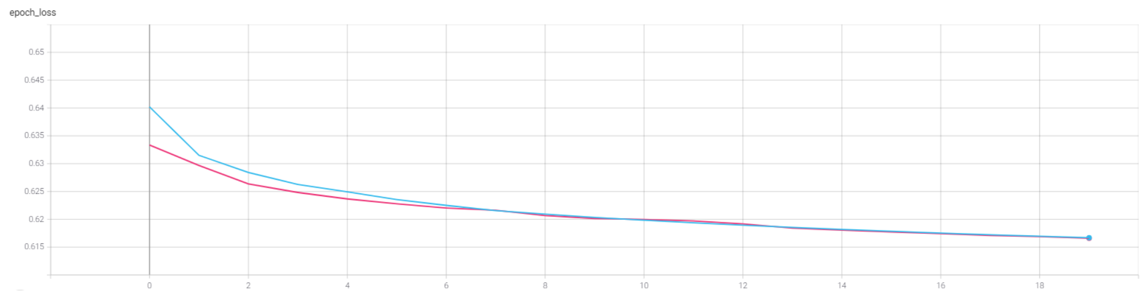


Figure 33. *Train and validation loss for pressure_diff+crackle using Xception*

As can be observed from figures above, the models' training and loss metrics were very similar when using these enhancement combinations. The combination of altitude and jerk_y reached 95.4% test accuracy and the combination of pressure_diff and crackle returned a slightly lower 87.2% test accuracy.

Direct comparison with state-of-the-art methods is difficult, since the dataset used in this thesis differs from most other related works. Works of Zarembo [7] and Nõmm [21] used the same dataset. Nonetheless, the comparison with other state-of-the-art methods shows that the proposed workflow produces results that are on par or superior to other state-of-the-art methods.

Diaz [5]	Kamran [6]	Pereira [22]	Ishii [8]	**Present Work**		
				AlexNet	LeNet5	Xception
94.44	99.22	90.38	79.00	95.44	85.54	**99.20**

Table 8. Comparison with related works

4. Conclusion

The primary goal of this thesis was to combine geometry based assessment and kinematic and pressure-parameter based approaches of PD prediction. Different augmentation methods of datasets and different CNN architectures were explored.

Enhancement and augmentation techniques used in this thesis were implemented. The dataset of 51 samples was first enhanced using different combinations of kinematic and pressure parameters and then augmented to roughly 30000 samples. The augmented dataset was used in the training of selected CNNs and showed results with high accuracy. Multiple different enhancement combinations were found, which produced high accuracy in PD prediction. Xception architecture produced the best result, with an enhancement combination using yaw_velocity for width and jerk_x for color producing 99.20% accuracy.

For future explorations other enhancement combinations of parameters could also be tested, since this thesis only used pressure and angle parameters for the width of the spiral and kinematic parameters for the coloration. Switching these around and/or mixing these parameters (e.g. using only kinematic parameters for width and color) would add another dimension to the settings.

Archimedean spiral tests were used in this thesis. Following the proposed workflow other tests such as handwriting, digit drawing and Lurias' alternating series tests could also be analysed and tested. Additional transformations such as brightness, noise, erosion etc. could also be used for augmentation. However some augmentation methods might not be as useful for some tests as they were with the Archimedean spirals.

Further adjustments could also be done to the models used in this thesis. More extensive hyper-parameter tuning could improve the results. Also, other CNN architectures such VGG19, Inception v3, ResNet50 etc. could be tested.

In conclusion, all the sub-problems stated were resolved and the goal of this thesis was met. Geometry based and kinematic and pressure-parameter based approaches were combined. The datasets were enhanced and augmented with selected methods and the evaluation of CNNs, which were trained on the datasets, produced superb results. The results achieved in this thesis show that the proposed workflow could be successfully implemented in this particular problem-domain and be applied in the analysis of other drawn and written tests.

Acknowledgments

I would like to thank my supervisors Sven Nõmm and Elli Valla for their help with writing this thesis, for keeping me on the right track and for keeping me constantly motivated. I would like to thank my family and my closest friends for their emotional support.

Bibliography

[1] Lorraine V Kalia and Anthony E Lang. “Parkinson’s disease”. In: *The Lancet* 386.9996 (2015), pp. 896–912. ISSN: 0140-6736. DOI: `https://doi.org/10.1016/S0140-6736(14)61393-3`. URL: `https://www.sciencedirect.com/science/article/pii/S0140673614613933`.

[2] Joseph Jankovic. “Parkinson’s disease: clinical features and diagnosis”. In: *Journal of Neurology, Neurosurgery & Psychiatry* 79.4 (2008), pp. 368–376.

[3] Jack J Chen. “Parkinson’s disease: health-related quality of life, economic cost, and implications of early treatment”. In: *The American journal of managed care* 16 Suppl Implications (Mar. 2010), S87–93. ISSN: 1088-0224. URL: `http://europepmc.org/abstract/MED/20297871`.

[4] S. Nomm et al. “Detailed Analysis of the Luria’s Alternating SeriesTests for Parkinson’s Disease Diagnostics”. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2018), pp. 1347–1352.

[5] Moises Diaz et al. “Sequence-based dynamic handwriting analysis for Parkinson’s disease detection with one-dimensional convolutions and BiGRUs”. In: *Expert Systems with Applications* 168 (2021), p. 114405. ISSN: 0957-4174. DOI: `https://doi.org/10.1016/j.eswa.2020.114405`. URL: `https://www.sciencedirect.com/science/article/pii/S0957417420310757`.

[6] Iqra Kamran et al. “Handwriting dynamics assessment using deep neural network for early identification of Parkinson’s disease”. In: *Future Generation Computer Systems* 117 (2021), pp. 234–244. ISSN: 0167-739X. DOI: `https://doi.org/10.1016/j.future.2020.11.020`. URL: `https://www.sciencedirect.com/science/article/pii/S0167739X20330442`.

[7] Sergei Zarembo et al. “CNN Based Analysis of the Luria’s Alternating Series Test for Parkinson’s Disease Diagnostics”. In: *Recent Challenges in Intelligent Information and Database Systems*. Ed. by Tzung-Pei Hong et al. Singapore: Springer Singapore, 2021, pp. 3–13. ISBN: 978-981-16-1685-3.

[8] Nobuyuki Ishii et al. “Spiral drawing: Quantitative analysis and artificial-intelligence-based diagnosis using a smartphone”. In: *Journal of the Neurological Sciences* 411 (2020), p. 116723. ISSN: 0022-510X. DOI: https://doi.org/10.1016/j.jns.2020.116723. URL: https://www.sciencedirect.com/science/article/pii/S0022510X20300599.

[9] Sven Nõmm et al. “Detailed Analysis of the Luria’s Alternating SeriesTests for Parkinson’s Disease Diagnostics”. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2018, pp. 1347–1352. DOI: 10.1109/ICMLA.2018.00219.

[10] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[11] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[12] Wes McKinney. “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[13] J. D. Hunter. “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[14] G. Bradski. “The OpenCV Library”. In: *Dr. Dobb’s Journal of Software Tools* (2000).

[15] Marcus D Bloice, Peter M Roth, and Andreas Holzinger. “Biomedical image augmentation using Augmentor”. In: *Bioinformatics* 35.21 (Apr. 2019), pp. 4522–4524. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz259. eprint: https://academic.oup.com/bioinformatics/article-pdf/35/21/4522/30330763/btz259.pdf. URL: https://doi.org/10.1093/bioinformatics/btz259.

[16] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[18] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[19] François Chollet. “Xception: Deep Learning with Depthwise Separable Convolutions”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1800–1807. DOI: 10.1109/CVPR.2017.195.

[20] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[21] S. Nomm et al. “Quantitative analysis in the digital Luria’s alternating series tests”. In: *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)* (2016), pp. 1–6.

[22] Clayton R. Pereira et al. “Convolutional Neural Networks Applied for Parkinson’s Disease Identification”. In: *Machine Learning for Health Informatics: State-of-the-Art and Future Challenges*. Ed. by Andreas Holzinger. Cham: Springer International Publishing, 2016, pp. 377–390. ISBN: 978-3-319-50478-0. DOI: 10.1007/978-3-319-50478-0_19. URL: https://doi.org/10.1007/978-3-319-50478-0_19.

Appendix 1 — Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks