Abstract

The emergence of autonomous vehicles signals a revolutionary shift in transportation, although these vehicles confront substantial challenges, primarily their capacity to accurately recognize and adapt to diverse road surfaces under various conditions. Leveraging recurrent neural networks (RNN) and Long Short-Term Memory (LSTM) models, this research delves into predicting road surface types using the comprehensive IMU sensor data from the "CareerCon 2019 - Help Navigate Robots" dataset. Aiming for a 97.53% accuracy rate, the study's pinnacle model, H-1, achieved a 71.39% test accuracy. Through rigorous experimentation, critical insights were obtained on performance nuances, overfitting, model complexity, and the potentials and pitfalls of deep architectures. These findings underscore the urgency for ongoing refinement and strategic application of neural networks in advancing autonomous vehicle safety.

Introduction

The evolution of transportation has taken a monumental leap with the introduction of autonomous vehicles. However, as futuristic and revolutionary as they may sound, these self-navigating marvels are confronted by challenges that could pose substantial risks. Among the myriad of challenges, the vehicle's competence in understanding and responding to varying road surface conditions, especially during adverse weather like rain or snow, stands paramount. A miscalculation or misinterpretation could be catastrophic, compromising the safety of its passengers and surroundings.

The core focus of this study lies in leveraging the power of recurrent neural networks (RNN) and the Long Short-Term Memory (LSTM) model. These computational structures are employed to predict road surface types, utilizing the Inertial Measurement Unit (IMU) sensor data harvested by a mobile robot. Such predictions are not just academic exercises; they hold the potential to significantly upgrade the safety protocols of autonomous vehicles, enabling them to navigate with heightened precision during various road conditions.

Drawing from the vast pool of data, our research hinges on the dataset procured from the Kaggle competition titled "CareerCon 2019 - Help Navigate Robots". This dataset boasts IMU sensor data gleaned from a mobile robot. Comprising 488,448 records from 10 distinct sensor channels with 128 measurements for each time series, this dataset provides an expansive playground for our model. The primary task here is to decipher which one of the nine floor types—ranging from soft_pvc, wood, to hard_tiles—the robot traverses, using pivotal sensor data metrics like acceleration and velocity.

It's not just about creating a functional model, but one that achieves an accuracy benchmark of 97.53%. Achieving this would not only catapult us into the top 5 scores of the competition but, more importantly, offer a solution that promises to elevate the navigation prowess of robots across a myriad of surfaces, ensuring their stability and reliability, preventing untimely "falls" during their operations.

Literature Review

The field of autonomous vehicles has been greatly influenced by significant contributions from various researchers and the advancements in deep learning techniques. This literature review highlights key publications that have shaped the understanding and development of deep learning techniques.

In 1943, McCulloch and Pitts proposed a logical calculus of interconnected neurons, known as the MCP model, to explore the generation of complex patterns in the brain (McCulloch and Pitts 1943). This model laid the foundation for artificial neural networks.

The widespread adoption of machine learning (ML) in research and applications has been evident in various domains, including text mining, spam detection, image classification, and multimedia concept retrieval (Alzubaidi et al. 2021). Alzubaidi et al. (2021) provided a comprehensive review of deep learning concepts, convolutional neural network (CNN) architectures, challenges, applications, and future directions. This review emphasizes the versatility of ML techniques.

Deep learning (DL) has revolutionized the field of pattern recognition by automating the learning of feature sets for various tasks (LeCun, Bengio, and Hinton 2015). LeCun et al. (2015) discussed the capabilities of DL in comparison to conventional ML methods. DL has the potential to significantly enhance the accuracy and efficiency of various tasks in automation.

Convolutional Neural Networks (CNNs) have been instrumental in achieving remarkable results in pattern recognition tasks, drawing inspiration from the visual system's structure (LeCun et al. 1998). LeCun et al. (1998) introduced CNNs, which employ error gradients to improve document recognition. The application of CNNs in object recognition has demonstrated their effectiveness in capturing and analyzing spatial features.

However, RNN architecture has a vanishing gradient problem, which means the model cannot hold the meanings of the words in long sequences. The issue in NLP is how computers understand the sequence of words as context; transformer architecture can solve this by using a self-attention mechanism that calculates weights for each word in sequences (Vaswani et al., 2017). Generative Pre-training (GPT) utilizes the transformer architecture but eliminates the Decoder. GPT-1 showed a significant improvement in accuracy by using a learning method called Generative Pretraining, a kind of semi-supervised learning (Radford et al., 2018).

Methods

Data Collection and Preparation

The dataset used for our experiment is sourced from the Kaggle competition titled "CareerCon 2019 - Help Navigate Robots". This data contains information acquired from Inertial Measurement Units (IMU sensors) gathered by a mobile robot navigating various floor surfaces within a university setting. Specifically, the robot's task is to correctly identify which among the nine possible floor types (namely, soft_pvc, wood, tiled, fine_concrete, hard_tiles_large_space, soft_tiles, carpet, and hard_tiles) it is traversing, based on the provided sensor data like acceleration and velocity. Notably, the data collection was facilitated by Heikki Huttunen, Francesco Lomio, Damoon Mohamadi, Kaan Celikbilek, Pedram Ghazi, and Reza Ghabcheloo from both the Department of Signal Processing and the Department of Automation and Mechanical Engineering at Tampere University, Finland.

Exploratory Data Analysis (EDA)

Dataset Overview: The dataset for this experiment comprises 487,680 data points, neatly organized into 13 columns. These columns are divided into two primary categories:

1. entification Columns (3)

- row_id: Acts as a unique identifier for each individual row in the dataset.
- series_id: An identifier for the measurement series. It acts as a foreign key to y_train/sample_submission.
- measurement_number: Represents the sequence of measurements within a series, allowing us to track and order the sequence of data entries.

2. Sensor Channels (10)

- Orientation Channels (4): These encode the current angles showcasing how the robot is oriented. The values are represented using quaternions (X, Y, Z, and W) rather than the conventional Euler Angles. This is due to the limitations of Euler Angles known as "gimbal lock." Gimbal lock occurs when the pitch angle approaches +/- 90 degrees, making it challenging to measure orientation. Quaternions, on the other hand, provide a robust alternative that doesn't suffer from this limitation. Although they are less intuitive and require more advanced mathematical comprehension, they offer a more accurate representation of three-dimensional rotations.

- orientation_X
- orientation_Y
- orientation_Z
- orientation_W

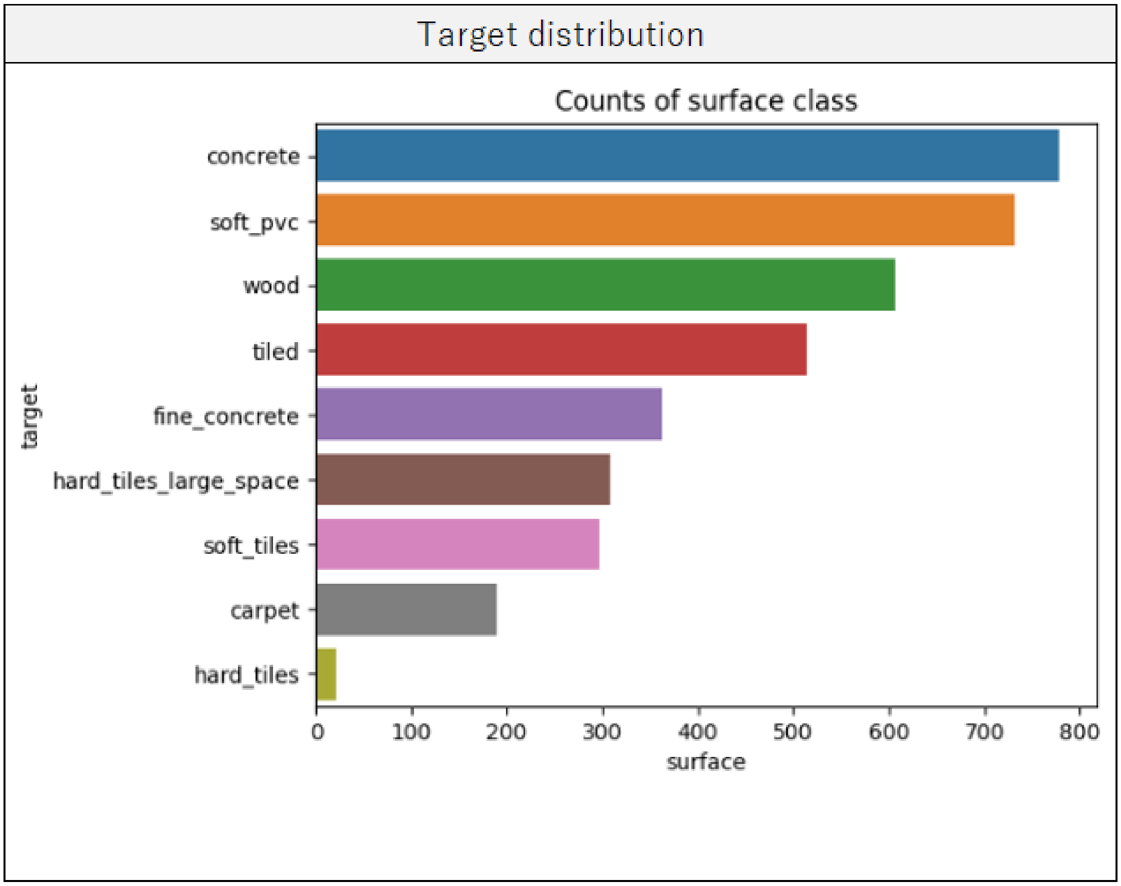
- Angular Velocity Channels (3): These channels describe both the angle and speed of motion of the robot.
  - angular_velocity_X
  - angular_velocity_Y
  - angular_velocity_Z
- Linear Acceleration Channels (3): These components convey how the speed of the robot changes over different intervals.
  - linear_acceleration_X
  - linear_acceleration_Y
  - linear_acceleration_Z

Traced data table looks like below.

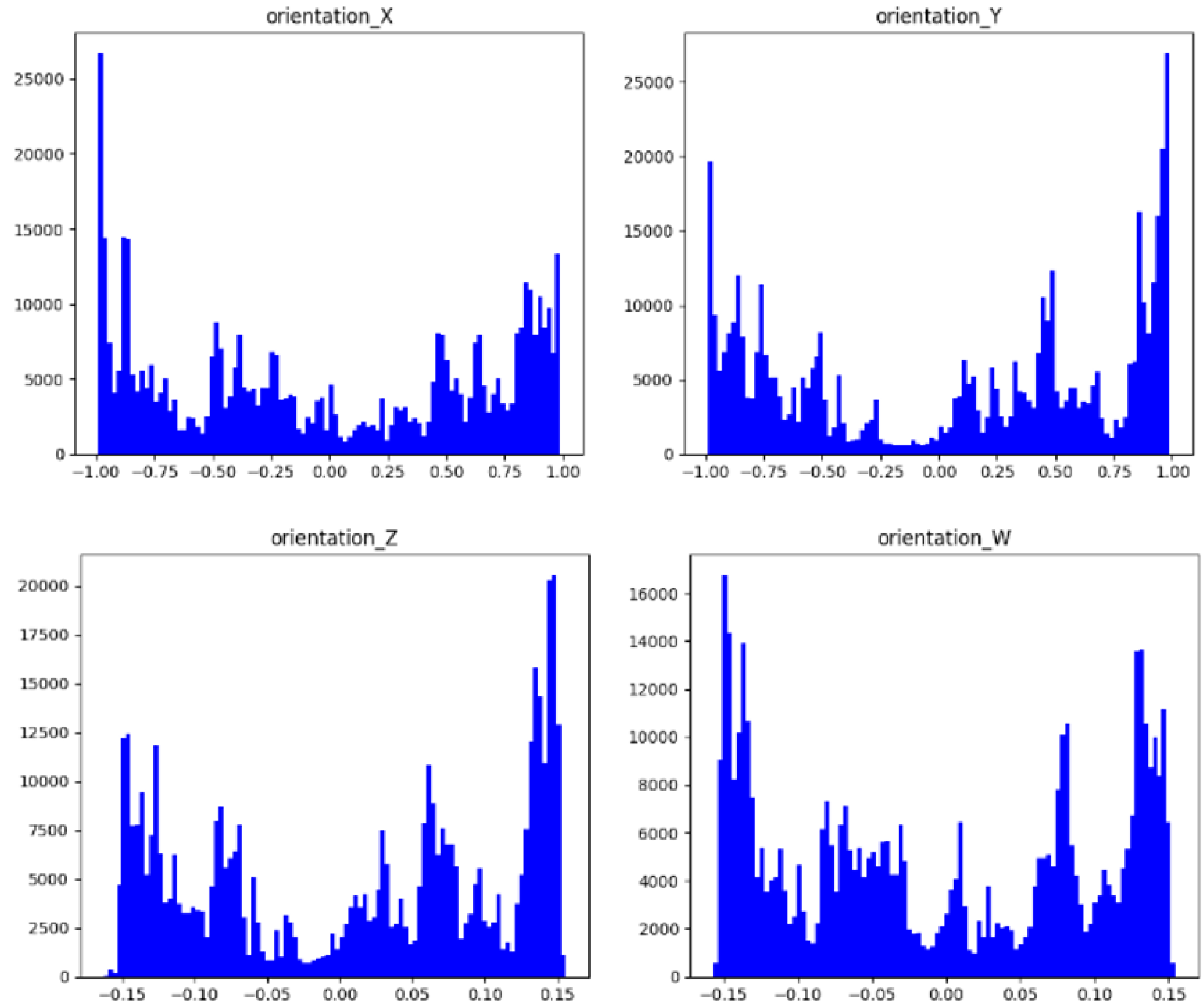
row_id	0	1	2
series_id	0_0	0_1	0_2
measurement_number	0	0	0
orientation_X	0	1	2
orientation_Y	-0.75853	-0.75853	-0.75853
orientation_Z	-0.63435	-0.63434	-0.63435
orientation_W	-0.10488	-0.1049	-0.10492

angular_velocity_X	-0.10597	-0.106	-0.10597
angular_velocity_Y	0.10765	0.067851	0.007275
angular_velocity_Z	0.017561	0.029939	0.028934
linear_acceleration_X	0.000767	0.003386	-0.005978
linear_acceleration_Y	-0.74857	0.33995	-0.26429
linear_acceleration_Z	2.103	1.5064	1.5922

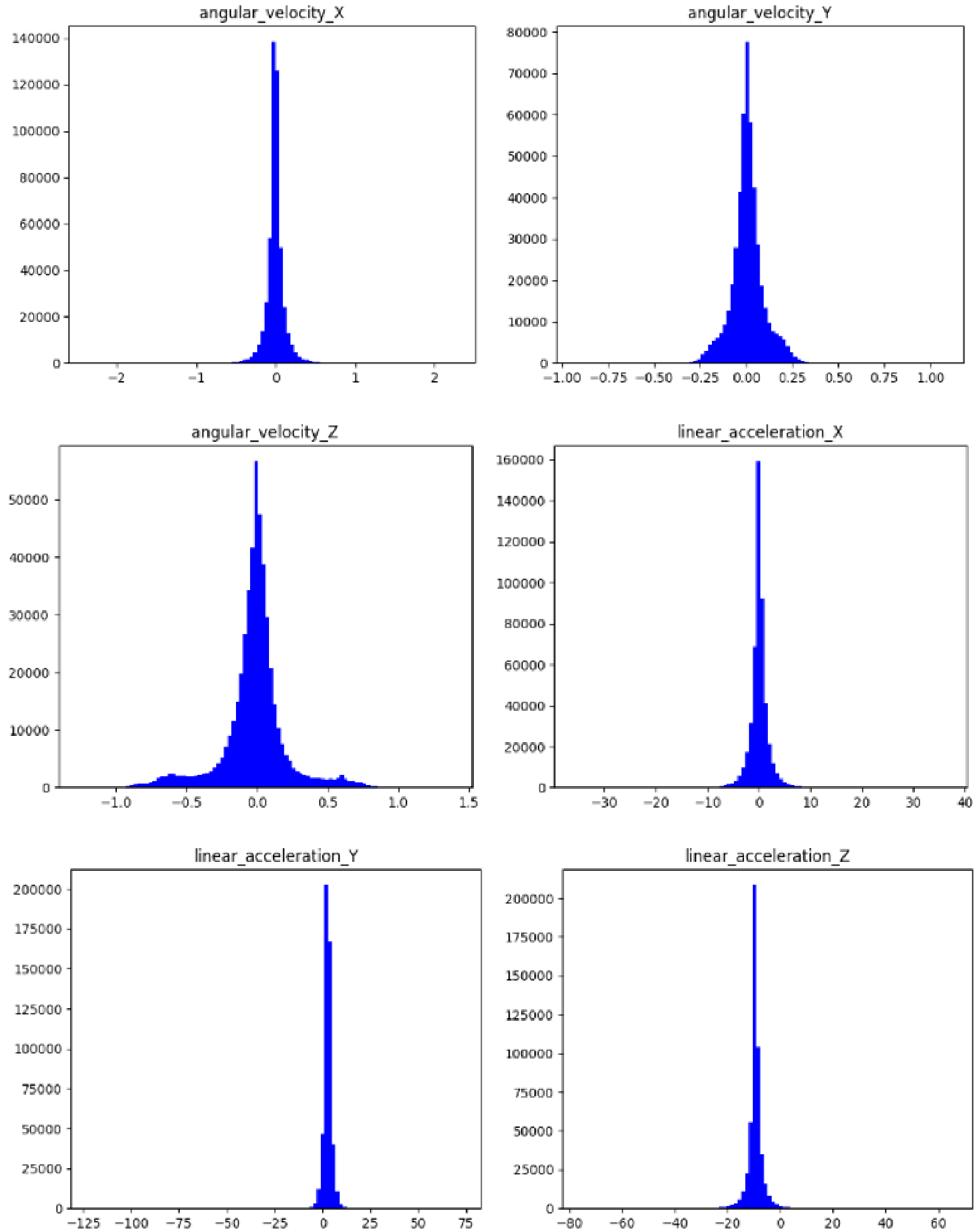
Dataset Composition: There are 3,810 unique series in the dataset, with each series containing 128 measurements. These measurements offer insights into the robot's orientation, angular velocity, and linear acceleration as it navigates across various surfaces. A notable challenge presented by the data is its imbalance. Certain classes (or floor surfaces) are underrepresented, being present in only a few series. This imbalance could introduce biases in the model if not addressed properly during the training phase.



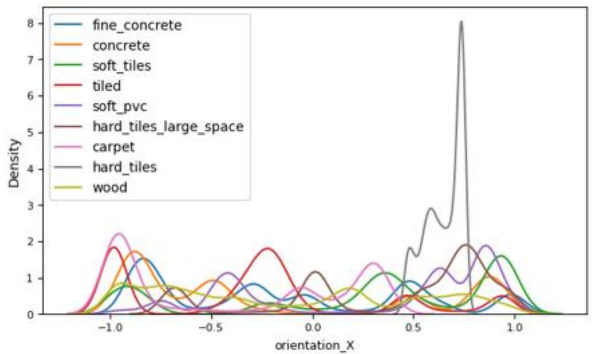
Velocity and Acceleration Distributions: Both velocity and acceleration data appear to follow a normal distribution. The orientation features, on the other hand, appear to have values normalized using the tanh function, indicating that the data might have undergone some preprocessing to transform it into a specific range or distribution.

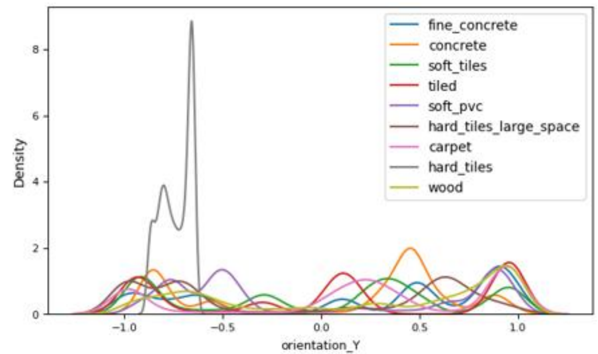


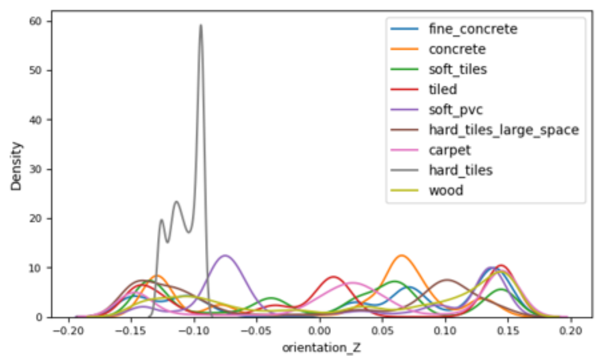
Sensor Data Analysis

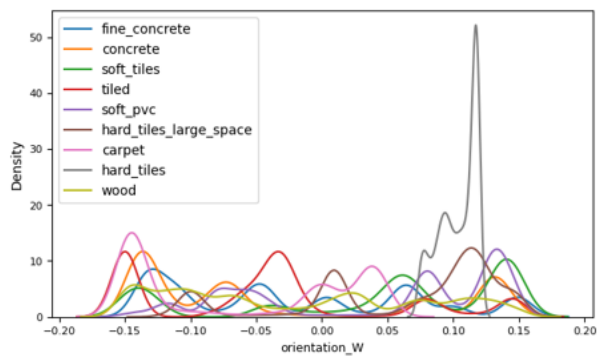


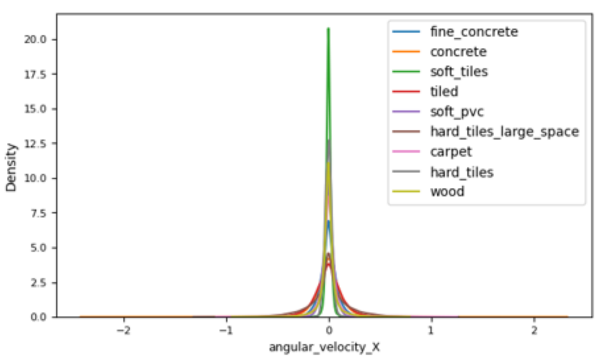
Surface Specific Distribution: On examining the distribution of each feature based on the target surfaces, it is apparent that the hard_tiles class exhibits distinct patterns, setting it apart from other surfaces.

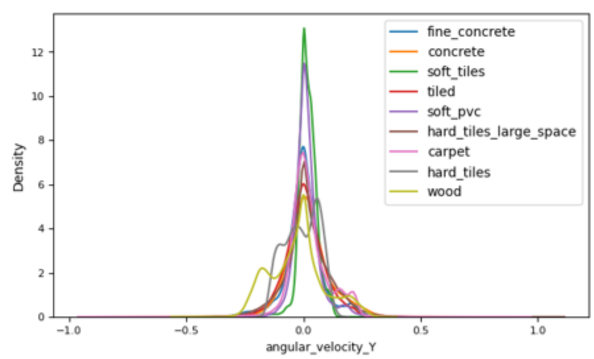


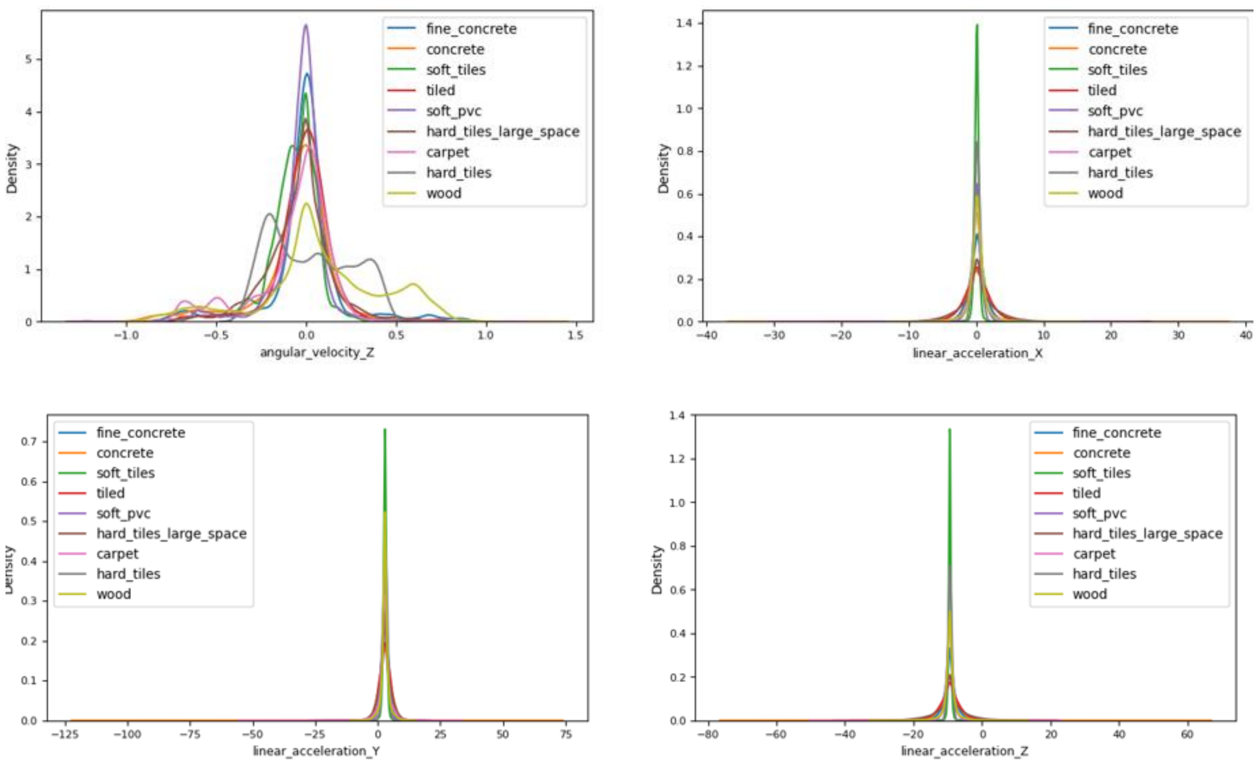




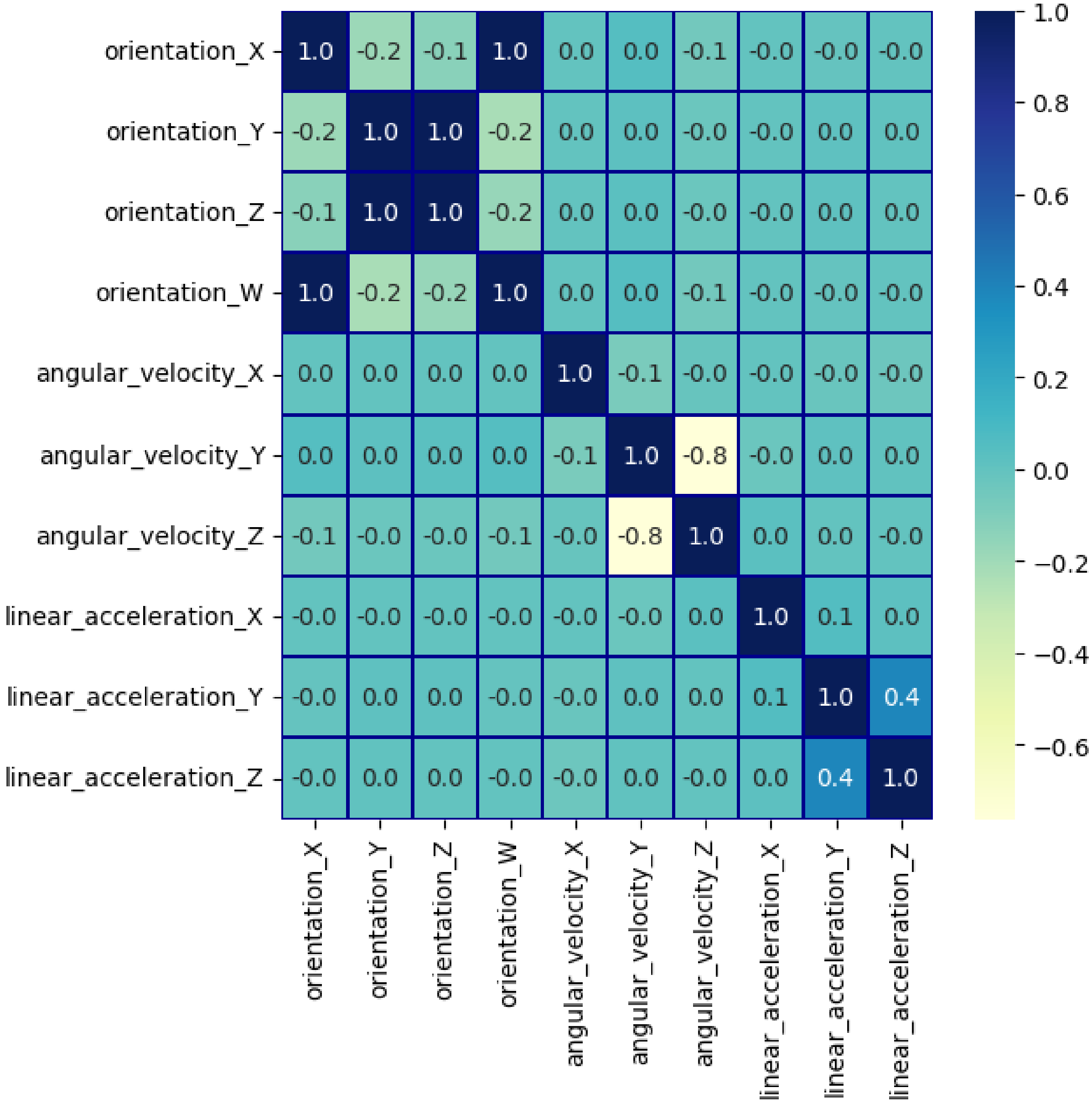




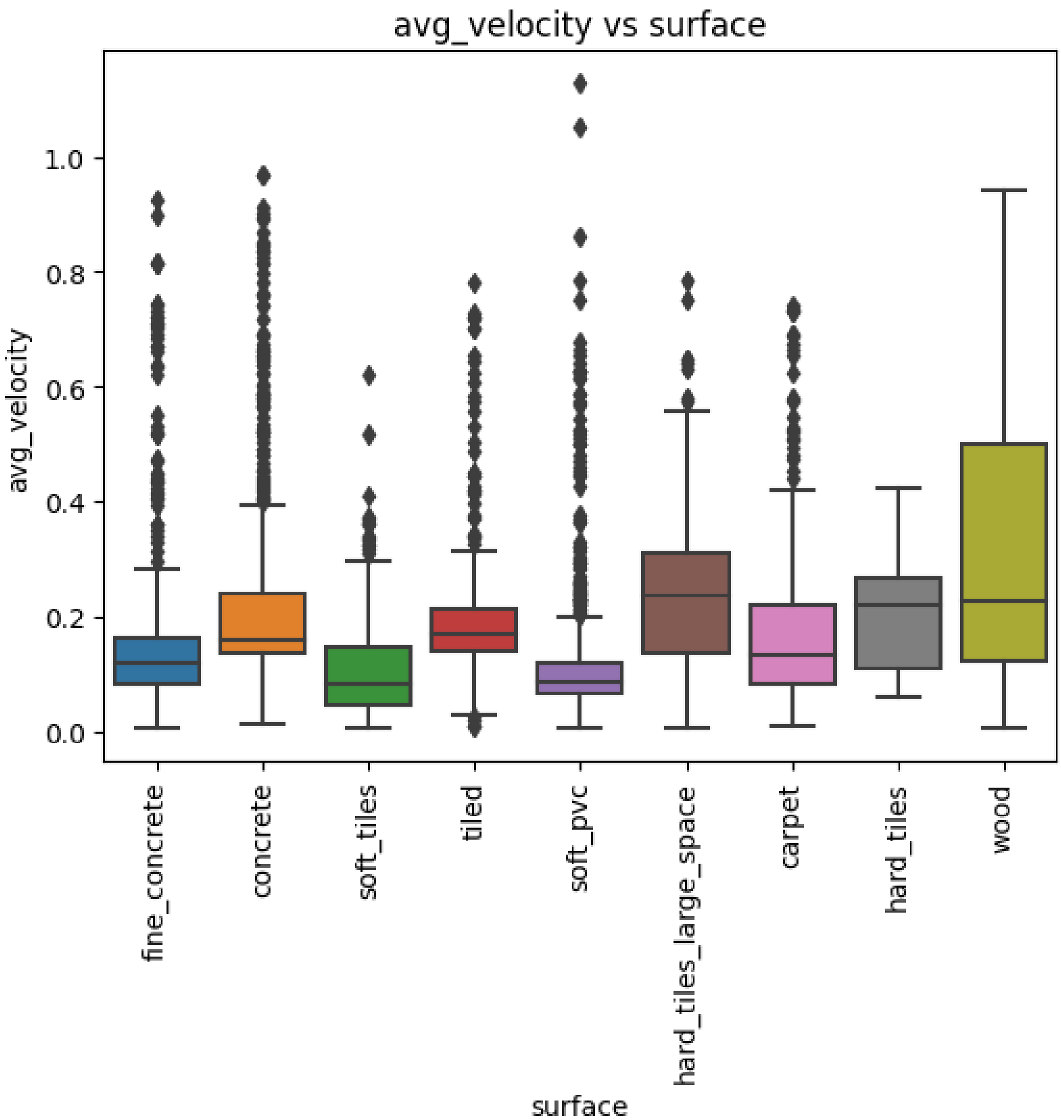




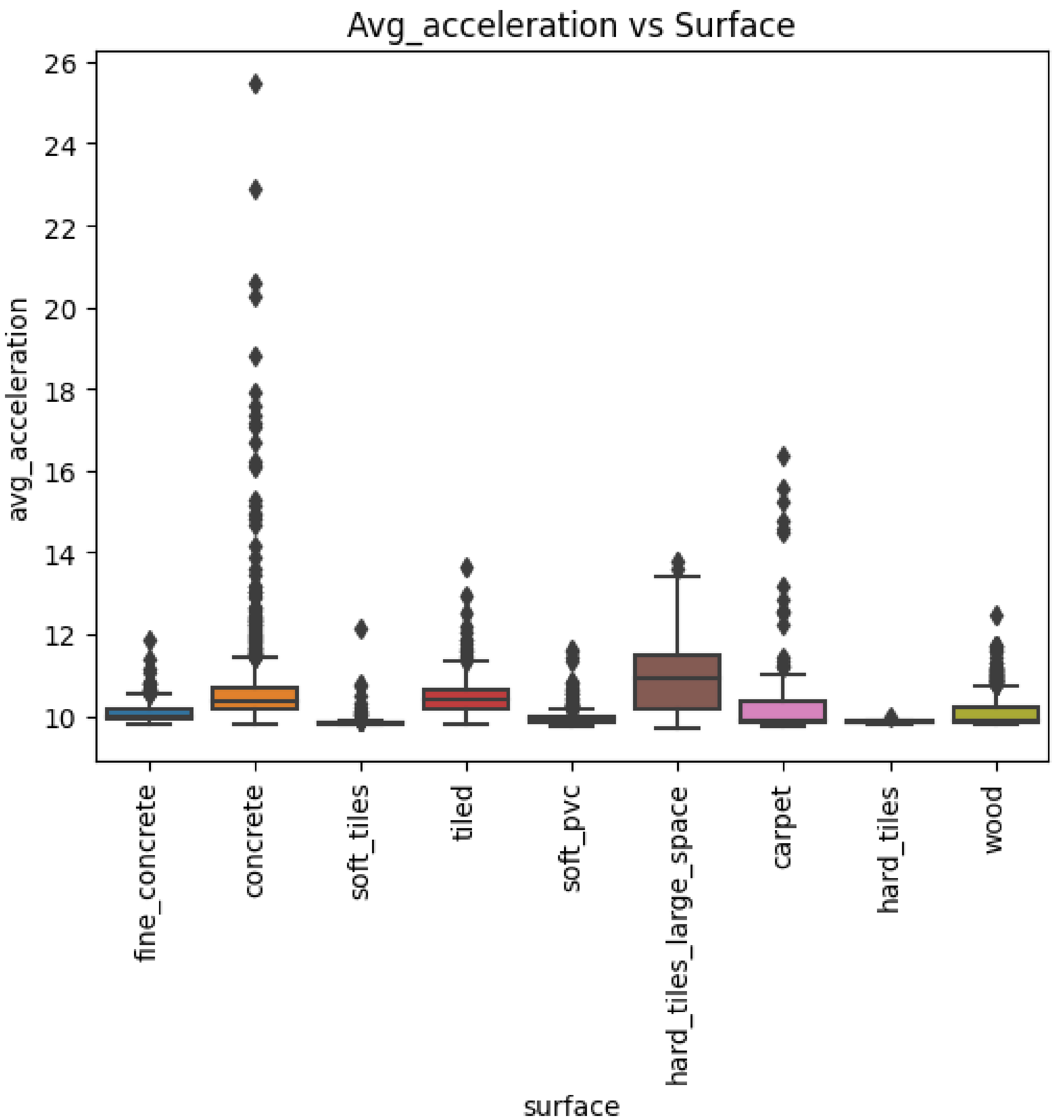
Feature Correlations: A perfect correlation (1.0) exists between orientation_X and orientation_W and between orientation_Z and orientation_Y. A strong inverse correlation (-0.8) is observed between angular_velocity_Z and angular_velocity_Y. A moderate positive correlation (0.4) exists between linear_acceleration_Y and linear_acceleration_Z.



Velocity Analysis: On assessing average velocities across different surfaces, some variations emerge. For instance, the robot seems to achieve a higher velocity on surfaces like wood compared to more restrictive surfaces like carpet.

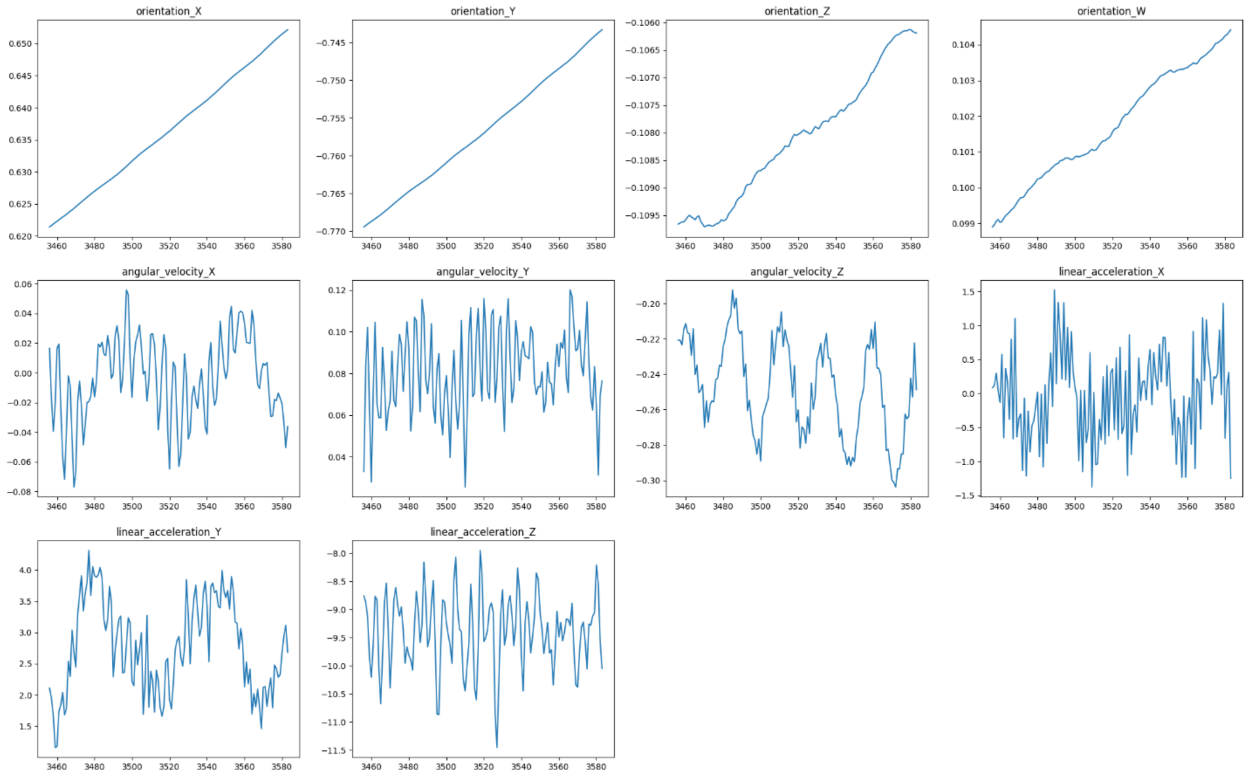


Acceleration Analysis: Similar patterns are noticed with acceleration. Different surfaces yield varying acceleration patterns, indicative of the surface texture and the robot's adaptability to it.

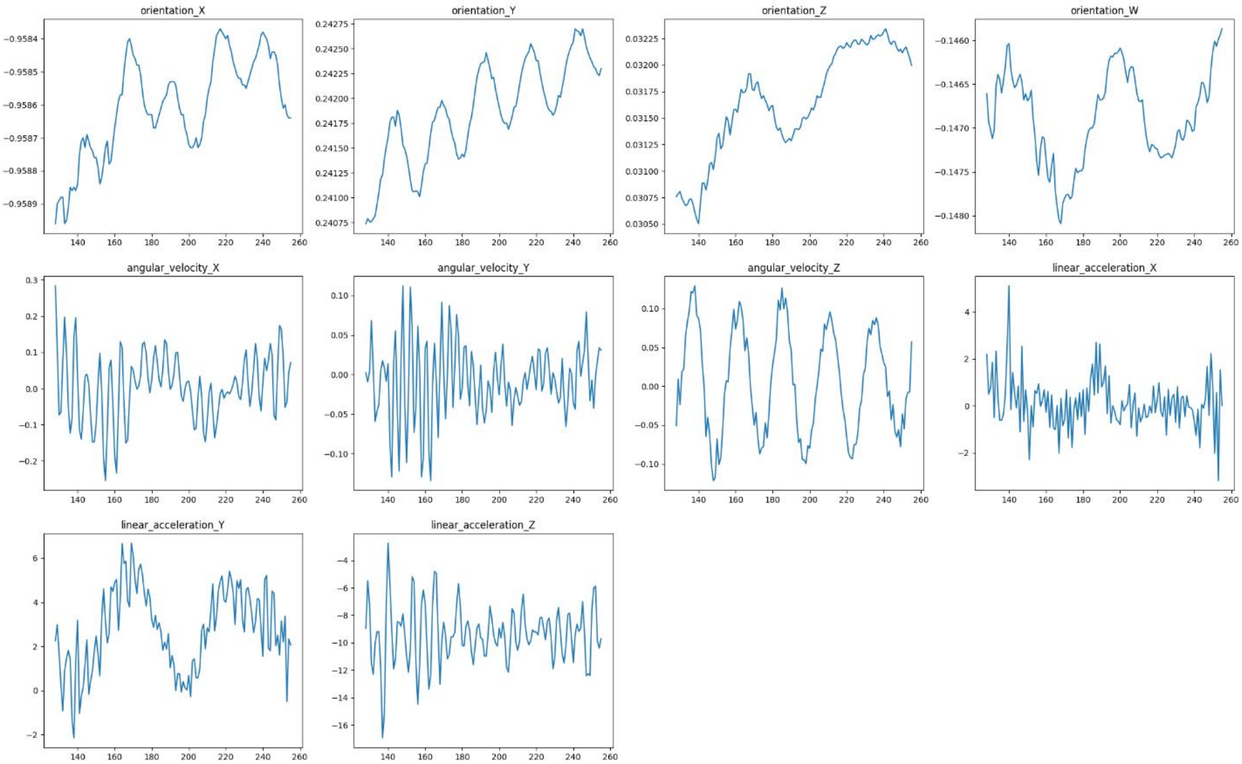


Sensor Behavior: A closer examination of how the robot's sensors react when navigating on hard tiles provides a representative example. The exact patterns and behaviors would depend on visualizations and more detailed data, but from the given data, we can infer that the robot's sensors respond differently based on the surface it's navigating.

On hard_tiles



On concrete



Data Preprocessing

Using Keras' train_test_split, the data was divided into training (1866 series), validation (801 series), and testing (1143 series) datasets. To maintain consistency across our experiments, we set a random state. Furthermore, the target distribution was ensured to be proportional across these datasets (See Appendix: Category distribution in train, validation, test dataset).

Research Design and Modeling

The focus of this research is on analyzing sequential data. Sequential data, especially time series data, presents unique challenges due to its temporal dependencies. Deep neural networks such as Recurrent Neural Networks (RNNs) including its variants like Simple RNN, LSTM, and

GRU, as well as 1d Convolutional Neural Networks (1d CNNs), are aptly suited for such tasks. The approach can be delineated into three main paths:

1. Types of Recurrent Neural Networks: Evaluating the different RNN structures like Simple RNN, LSTM, and GRU.
2. Hyperparameter Tuning and Regularization Techniques: Adjusting network parameters for optimal performance while ensuring the network does not overfit or underfit.
3. Network Architecture: Examining the inherent structure of the network, including the number of layers, nodes, and connections.

The chosen network topologies for investigation are as follows:

- Simple RNN: RNNs are designed to recognize patterns in sequences of data, such as text, genomes, handwriting, the spoken word, or numerical times series data emanating from sensors, stock markets, and government agencies. They possess 'memory' which captures information about what has been calculated so far. Bidirectional RNNs (BRNN) enhance the traditional RNN structure by taking into consideration future data for improved accuracy. This is akin to predicting a word in a sentence by not just considering previous words, but also the ones that follow.
- LSTM: LSTM networks are a type of RNN that uses special units in addition to standard units. These special units include gates which manage the flow of information. They can remember patterns over long durations and are less susceptible to the vanishing gradient problem.
- GRU:GRU is another RNN variant which often outperforms Simple RNN. It combines the forget and input gates into a single “update gate”. It also merges the cell state and hidden state,

thus generally being lighter and faster than LSTM. However, LSTMs are more effective for longer sequences.

- 1d CNN: Originally designed for image processing, Convolutional Neural Networks (CNNs) have found applications in Natural Language Processing (NLP) as well. Here, they act as feature extractors from sentences, capturing local semantic features.

Regularization is essential to prevent overfitting in deep learning models. Some of the techniques include:

- Dropout: A technique where certain neurons are randomly 'dropped-out' or deactivated during training, promoting a more generalized model.
- Early Stopping: Training is halted once the model's performance ceases to improve on a validation set. This ensures that the model does not overfit.
- Weight Decay (L1 & L2 Regularization): Penalizes large weights in the model by adding a term to the loss function. While L1 adds the absolute values of the weights, L2 adds the squared values, driving the model towards smaller weights.

Summary EXPERIMENT modeling methods are below.

Experiment	Model number	Experiment Description
B	B-1	simple RNN
B	B-2	Bidirectional simple RNN
B	B-3	GRU
B	B-4	Bidirectional GRU
B	B-5	LSTM
B	B-6	Bidirectional LSTM
C	C-1~5	GRU with different number of nodes(32, 64, 128, 256, 512)
D	D-1~5	GRU with different number of nodes(32, 64, 128, 256, 512) and regularization(early stopping)
E	E-1~3	GRU with 256 nodes and regularization(early stopping / dropout)
F	F-1~3	GRU with 256 nodes and regularization(early stopping / reccurent dropout)
G	G-1~3	GRU with 256 nodes and regularization(early stopping / reccurent dropout / weight decay)
H	H-1	2 layer GRU with 256 nodes and regularization(early stopping / reccurent dropout)
I	I-1	1dCNN+ GRU with 256 nodes and regularization(early stopping / reccurent dropout)
J	J-1	3 layer GRU with 256 nodes and regularization(early stopping / reccurent dropout)

Implementation and Programming

The implementation is carried out using the Keras library, a high-level deep learning API written in Python, running on top of the TensorFlow platform. Keras offers simplicity, flexibility, and power, making it an ideal choice for developing and shipping machine learning solutions with high iteration velocity. Matplotlib and Seaborn are utilized for data visualization, enabling the generation of aesthetically pleasing and statistically sophisticated visualizations to aid in result interpretation and analysis.

Results

Result Summary

Experiment	Model Num	Description	Hidden layer nodes	Regularization	Epoch	Train Accuracy	Validation Accuracy	Test Accuracy	Train Loss	Validation Loss	Test Loss	Train Time	Processing Unit
B	B-1	simple RNN	10		30	0.3650	0.3208	0.3412	1.6743	1.7457	1.7233	1min 15s	CPU
B	B-2	Bidirectional simple RNN	10		30	0.3741	0.3533	0.3473	1.5628	1.6230	1.6661	2min	CPU
B	B-3	GRU	10		30	0.4753	0.4682	0.4453	1.4069	1.4467	1.4629	2min 52s	CPU
B	B-4	Bidirectional GRU	10		30	0.4796	0.4482	0.4462	1.3882	1.5341	1.5341	4min 27s	CPU
B	B-5	LSTM	10		30	0.4421	0.4457	0.4331	1.4538	1.6786	1.5664	2min 9s	CPU
B	B-6	Bidirectional LSTM	10		30	0.3864	0.3708	0.3683	1.6003	1.6256	1.6932	4min 15s	CPU
C	C-1	GRU with different number of nodes(**32**, 64, 128, 256, 512)	32		30	0.5975	0.5581	0.5503	1.0849	1.2218	1.2810	12min 44s	CPU
C	C-2	GRU with different number of nodes(32, **64**, 128, 256, 512)	64		30	0.6715	0.6292	0.5984	0.9227	1.1078	1.2143	12min 30s	CPU
C	C-3	GRU with different number of nodes(32, 64, **128,** 256, 512)	128		30	0.7706	0.6292	0.6404	0.6443	1.0979	1.1717	12min 22s	CPU
C	C-4	GRU with different number of nodes(32, 64, 128, **256**, 512)	256		30	0.8542	0.6754	0.6684	0.4313	1.0461	1.2157	12min 30s	CPU
C	C-5	GRU with different number of nodes(32, 64, 128, 256, **512**)	512		30	0.9121	0.6904	0.6877	0.2646	1.2445	1.2857	12min 23s	CPU
D	D-1	GRU with different number of nodes(**32**, 64, 128, 256, 512) and regularization(early stopping)	32	Early Stopping	24	0.5450	0.5156	0.5022	1.2198	1.3602	1.3755	2min 17s	CPU
D	D-2	GRU with different number of nodes(32, **64**, 128, 256, 512) and regularization(early stopping)	64	Early Stopping	20	0.6131	0.5643	0.5363	1.0554	1.2381	1.3065	2min 15s	CPU
D	D-3	GRU with different number of nodes(32, 64, **128**, 256, 512) and regularization(early stopping)	128	Early Stopping	24	0.6919	0.6055	0.5704	0.8259	1.1566	1.2585	5min 4s	CPU
D	D-4	GRU with different number of nodes(32, 64, 128, **256**, 512) and regularization(early stopping)	256	Early Stopping	23	0.7690	0.6330	0.6133	0.6279	1.1471	1.2186	11min 22s	CPU
D	D-5	GRU with different number of nodes(32, 64, 128, 256, **512**) and regularization(early stopping)	512	Early Stopping	15	0.7138	0.6242	0.5731	0.7756	1.1039	1.2054	23min 45s	CPU

Experiment	Model Num	Description	Hidden layer nodes	Regularization	Epoch	Train Accuracy	Validation Accuracy	Test Accuracy	Train Loss	Validation Loss	Test Loss	Train Time	Processing Unit
E	E-1	GRU with 256 nodes and regularization(early stopping / dropout = 0.3)	256	Early Stopping/Dropout(0.3)	6	0.4175	0.3321	0.3596	1.4987	1.7735	1.7391	1min 53s	CPU
E	E-2	GRU with 256 nodes and regularization(early stopping / dropout = 0.5)	256	Early Stopping/Dropout(0.5)	6	0.4202	0.2185	0.2012	1.5146	2.2395	2.1953	1min 52s	CPU
E	E-3	GRU with 256 nodes and regularization(early stopping / dropout = 0.7)	256	Early Stopping/Dropout(0.7)	7	0.4212	0.1948	0.2196	1.5177	2.2786	2.1954	2min 10s	CPU
F	F-1	GRU with 256 nodes and regularization(early stopping / reccurent dropout = 0.3)	256	Early Stopping / Reccurent Dropout(0.3)	22	0.7267	0.6617	0.6535	0.7497	0.9994	1.0369	23min 43s	CPU
F	F-2	GRU with 256 nodes and regularization(early stopping / reccurent dropout = 0.5)	256	Early Stopping / Reccurent Dropout(0.5)	30	0.7417	0.6330	0.6352	0.7133	1.0649	1.0686	32min 15s	CPU
F	F-3	GRU with 256 nodes and regularization(early stopping / reccurent dropout = 0.7)	256	Early Stopping / Reccurent Dropout(0.7)	30	0.7181	0.6742	0.6387	0.7815	0.9273	1.0235	32min 10s	CPU
G	G-1	GRU with 256 nodes and regularization(early stopping / reccurent dropout = 0.3 / weight decay = L1)	256	Early Stopping / Reccurent Dropout(0.3) / L1 reguralization	30	0.5857	0.5019	0.5153	1.4366	1.6236	1.6640	36min 51s	CPU
G	G-2	GRU with 256 nodes and regularization(early stopping / reccurent dropout = 0.3 / weight decay = L2)	256	Early Stopping / Reccurent Dropout(0.3) / L2 reguralization	20	0.5820	0.5718	0.5599	1.2602	1.3233	1.3585	24min 33s	CPU
G	G-3	GRU with 256 nodes and regularization(early stopping / reccurent dropout = 0.3 / weight decay = L1-L2)	256	Early Stopping / Reccurent Dropout(0.3) / L1-L2 reguralization	30	0.6066	0.5443	0.5372	1.4135	1.5752	1.6187	36min 47s	CPU
H	H-1	2 layer GRU with 256 nodes and regularization(early stopping / reccurent dropout = 0.3)	256, 256	Early Stopping / Reccurent Dropout(0.3)	28	0.8842	0.7341	0.7139	0.3423	0.9922	1.0905	1h 40min 1s	CPU
I	I-1	1dCNN+ GRU with 256 nodes and regularization(early stopping / reccurent dropout = 0.3)	150, 256	Early Stopping / Reccurent Dropout(0.3)	20	0.8499	0.7278	0.6728	0.4109	0.9607	1.1126	16min 43s	CPU
J	J-1	3 layer GRU with 256 nodes and regularization(early stopping / reccurent dropout = 0.3)	256, 256, 256	Early Stopping / Reccurent Dropout(0.3)	13	0.6613	0.6355	0.6098	0.9183	1.0633	1.1169	1h 8min 46s	CPU

Experiment B: Comparative Analysis of Different RNN Types

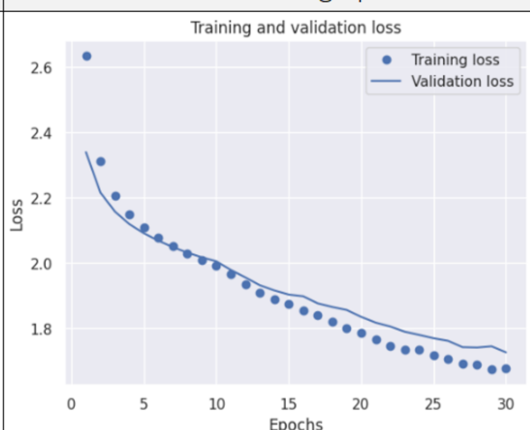
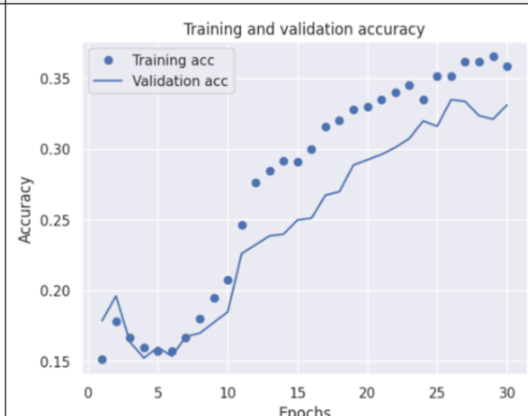
Experiment B-1: Simple RNN

In our study, we ventured into a comparative analysis of various RNN types, with Experiment B specifically focusing on the Simple RNN.

The Simple RNN model was architecturally designed with an input layer, followed by a 10-unit Simple RNN layer, and culminated with an output layer. This output layer employed a Softmax activation function, optimized for multi-class classification scenarios. The "sparse_categorical_crossentropy" was our chosen loss function, with the Adam optimizer guiding the learning process. Unless explicitly mentioned, the model relied on default hyperparameter settings.

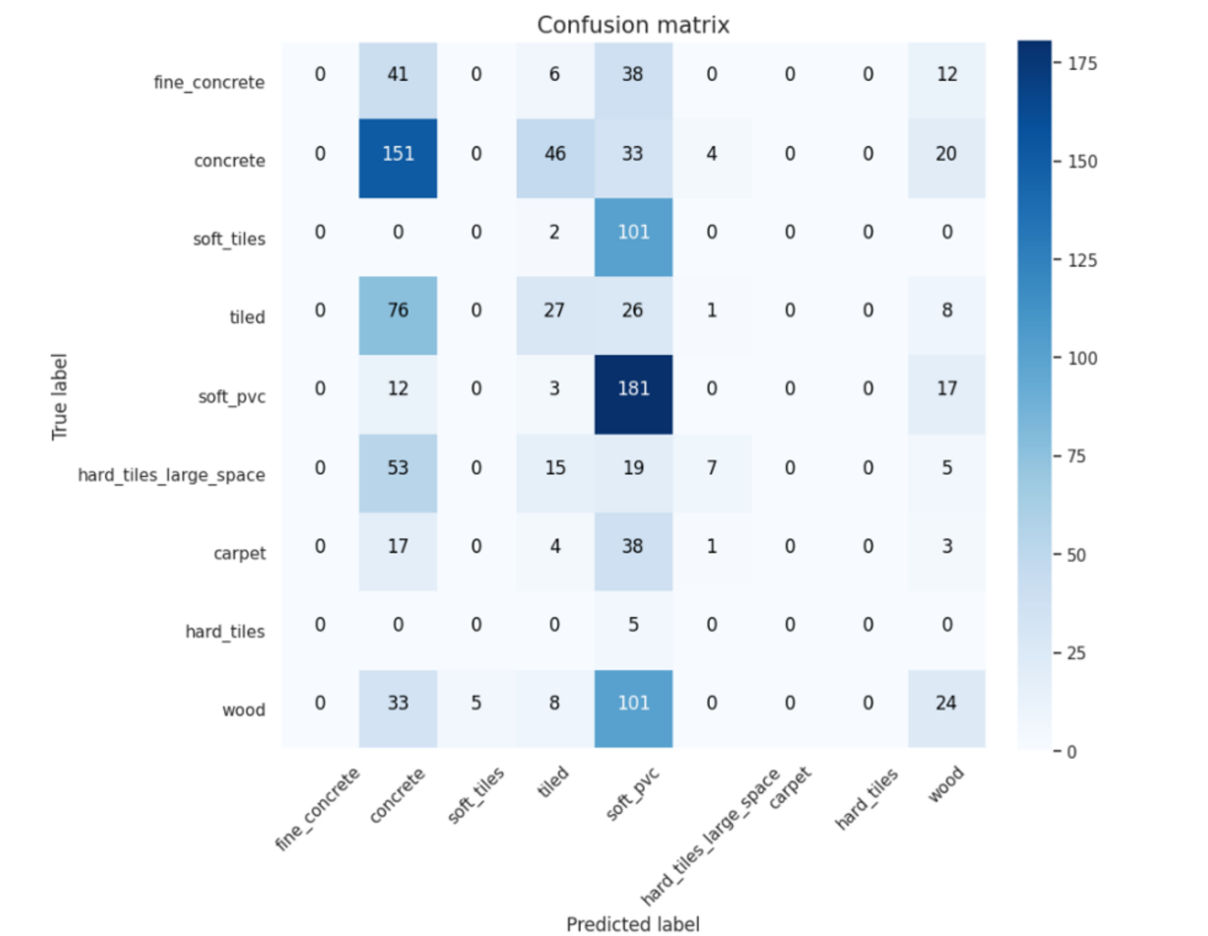
After subjecting the model to 30 epochs of training, we gauged its performance metrics. The results were modest, with a training accuracy of 0.365, validation accuracy standing at 0.3208, and a test accuracy of 0.3412. This entire training process lasted for a brief duration of 1 minute and 15 seconds.

To further understand the model's progression and performance, we incorporated visual insights. Using Matplotlib, we illustrated the ebbs and flows of both the model's accuracy and loss across epochs. A noticeable pattern emerged wherein the accuracy improvement seemed to plateau after the 15th epoch.

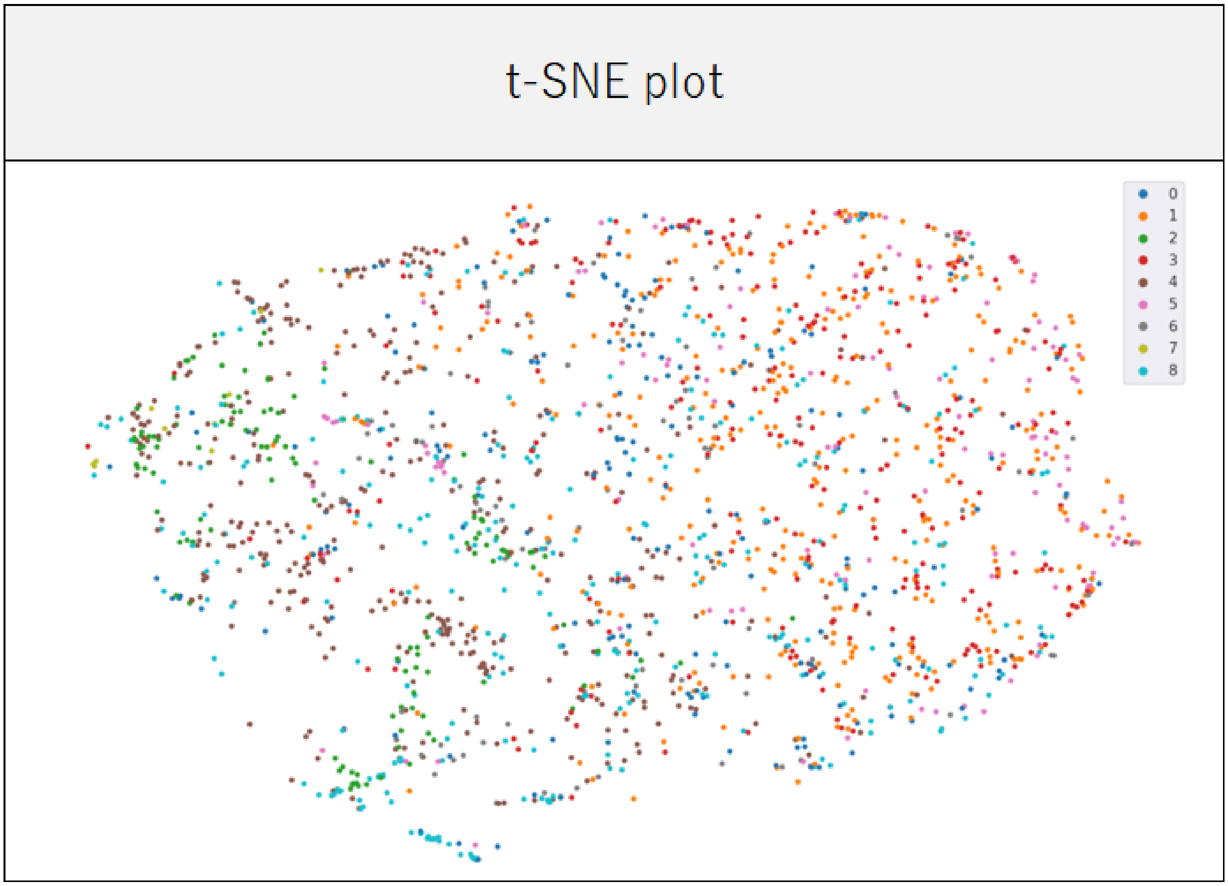
ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
B-1		

The confusion matrix, a pivotal tool to assess classification models, showcased a deviation from the ideal. Instead of a pronounced diagonal indicating accurate predictions, our matrix hinted at the model's inconsistencies.

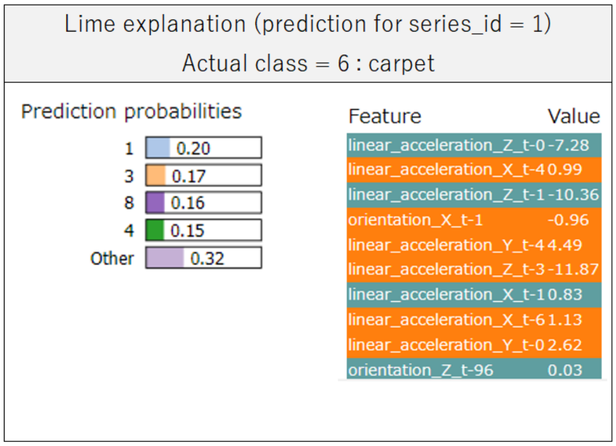
The heat map of the confusion matrix

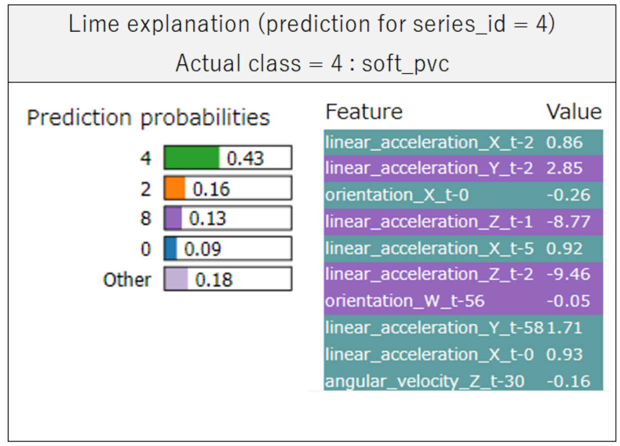


In our exploration, we also used t-SNE for visualization. This method highlighted weak categorical trends in activation values. Categories, specifically those colored blue (like 0 and 8), predominantly positioned themselves to the left. Conversely, the red and orange categories (like 1a and 3) were more oriented towards the right.



Aiming for a deeper introspection of our model's behavior, we utilized LIME (Local Interpretable Model-agnostic Explanations). One of our key observations came from series_id = 1. The model's prediction diverged from the actual class "6: carpet" and leaned towards "1", influenced majorly by the 'linear_acceleration' feature. A similar influence of 'linear_acceleration' was observed for series_id = 4, where both the actual and model's prediction aligned at "4: soft_pvc".





In conclusion, this initial analysis with a simple RNN model was a learning experience. The results, though not groundbreaking, set the stage for our future endeavors. We aim to delve deeper into various RNN architectures, seeking a model that not only resonates with our data but also promises enhanced predictive powers.

Experiment B-2: Bidirectional Simple RNN

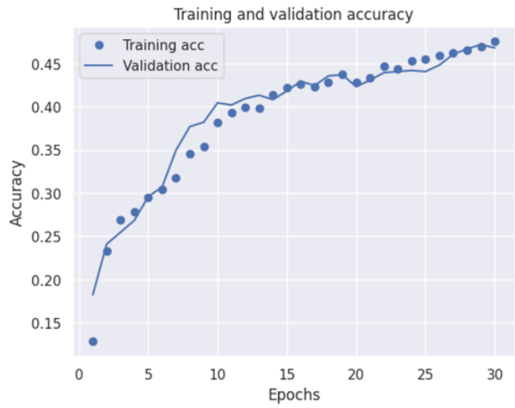
In our continued exploration, we employed a bidirectional version of the simple RNN. This model, constructed with an input layer, followed by a bidirectional simple RNN layer, and concluding with an output layer, held 10 units in the simple RNN layer. The Softmax activation function, the Adam optimizer, and the "sparse_categorical_crossentropy" loss function were once again integral to our model. The training phase spanned 30 epochs and yielded a train accuracy of 0.3741, a validation accuracy of 0.3533, and a test accuracy of 0.3473. Notably, the training time extended from 1 minute to 2 minutes, but this elongation didn't translate to significant accuracy enhancements. The outcomes of the confusion matrix and t-SNE plot mirrored those of Experiment B-1, with further details provided in the appendix under Experiment B-2.

Experiment B-3: GRU (Gated Recurrent Unit)

In our pursuit of refining neural network architectures, we directed our attention towards the Gated Recurrent Unit (GRU). A model was constructed, consisting of an input layer, followed by a GRU layer, and culminating with an output layer. Specifically, the GRU layer was designed with 10 units. For the output layer, the Softmax activation function was employed. Optimization was facilitated by the Adam optimizer and the model was trained using the "sparse_categorical_crossentropy" loss function. All other hyperparameter settings were retained at their default values.

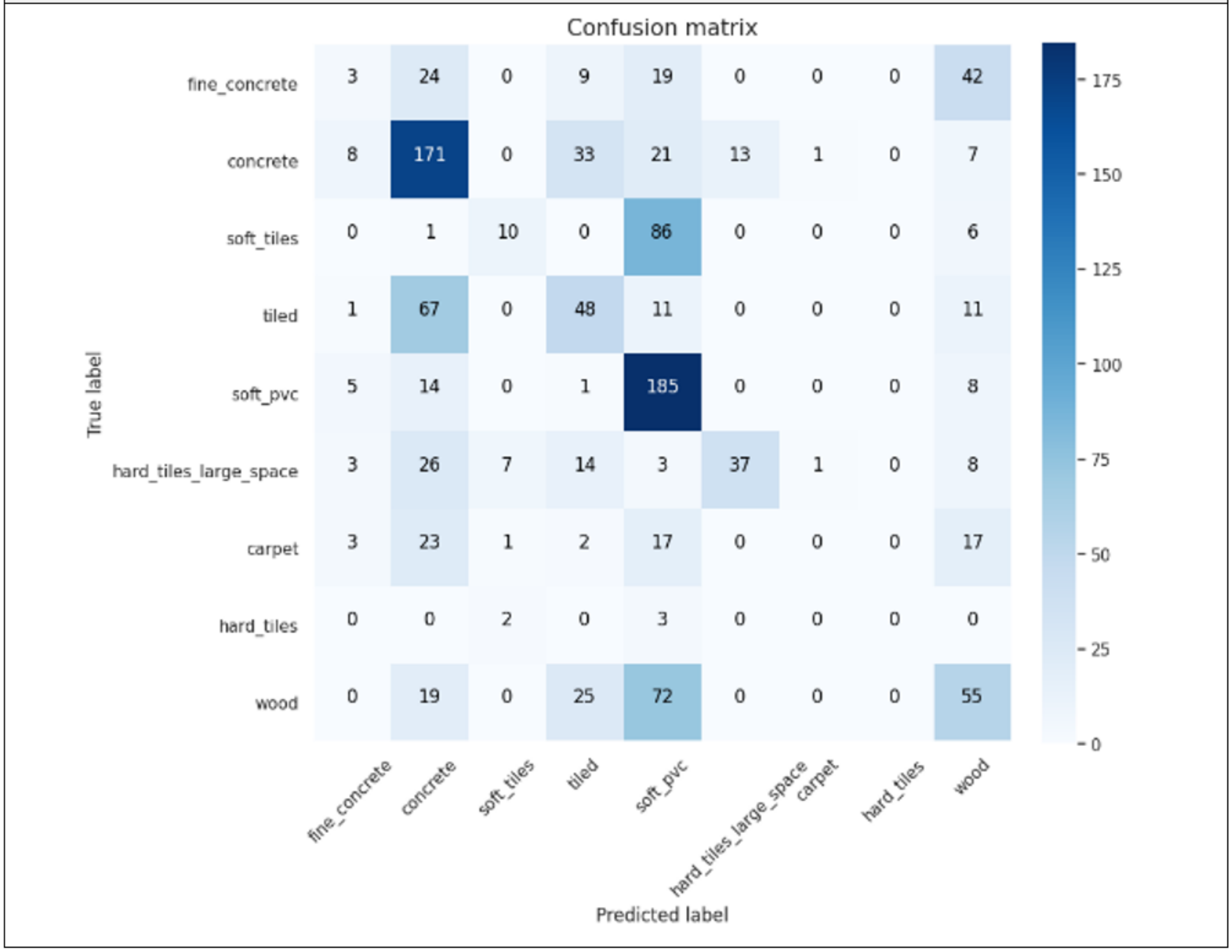
After 30 training epochs, the train accuracy reached 0.4753, while the validation and test accuracies touched 0.4682 and 0.4453, respectively. A comparative analysis revealed that the accuracy of this GRU model was significantly superior to that of a simple RNN. The GRU, as introduced by Cho, et al. in 2014, is strategically designed to address the vanishing gradient problem inherent to traditional recurrent neural networks. Intriguingly, the GRU can be perceived as an LSTM variant, given their similar design philosophies. In certain scenarios, both architectures have showcased commendable performances.

Visual insights from Matplotlib revealed an accuracy augmentation deceleration post the 10th epoch. Yet, the trend's consistency insinuates potential uplifts with hyperparameter tweaks.

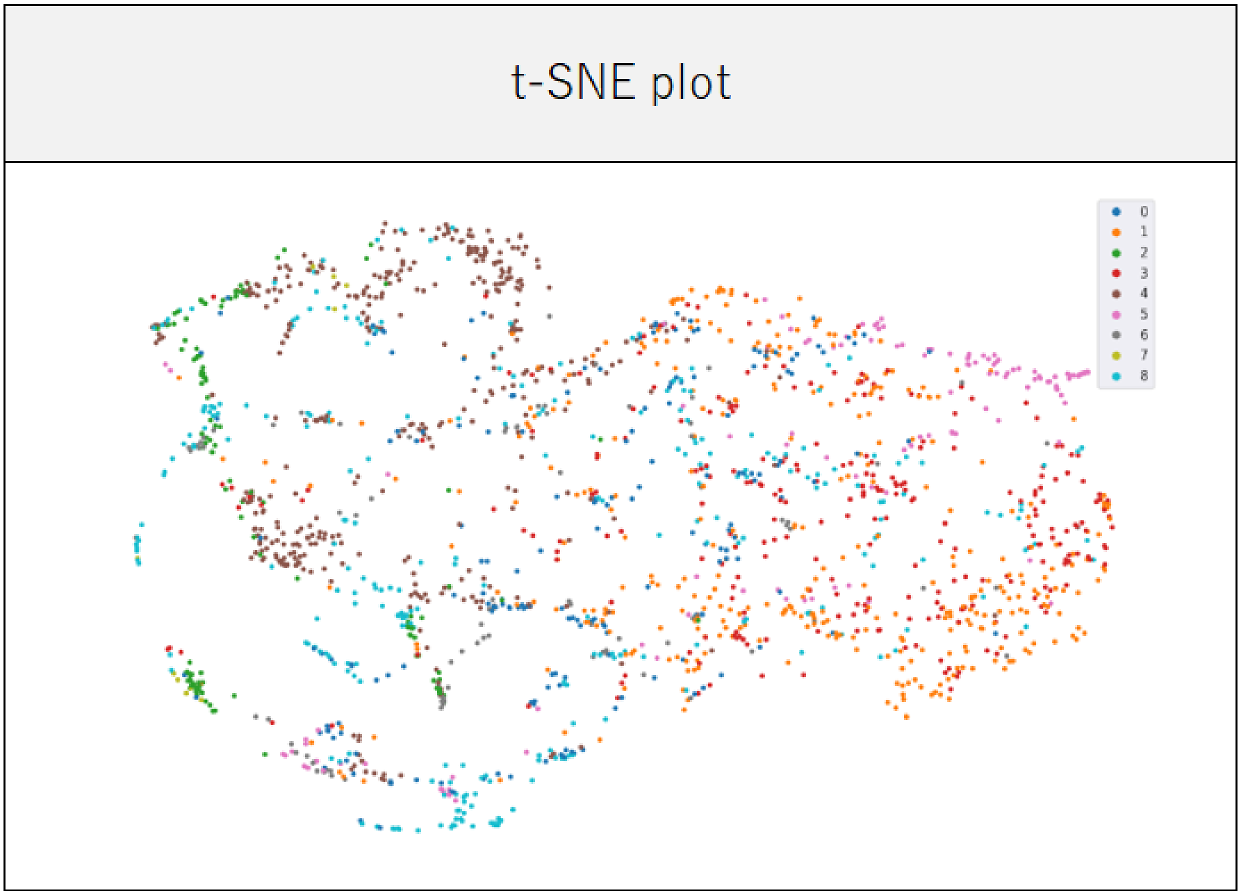
ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
B-3		

The confusion matrix shown areas for improvement. Yet, compared to a simple RNN model, a illumination of the diagonal cells was evident, suggesting a better alignment of predictions with actual values.

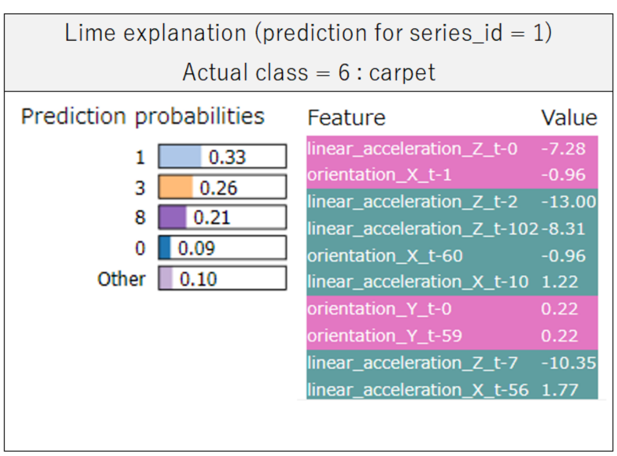
The heat map of the confusion matrix

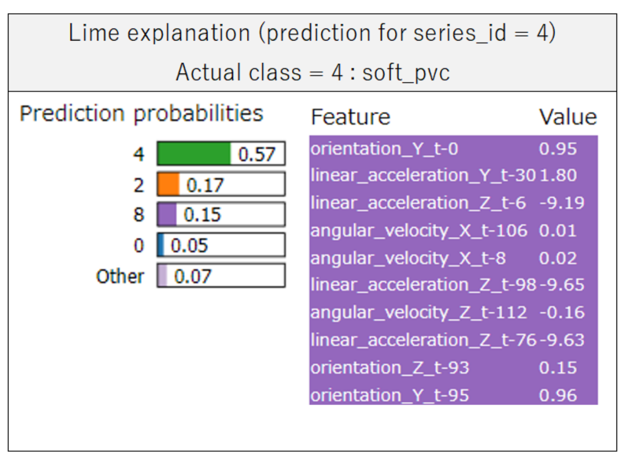


Observations exhibited a trend akin to that from model B-1. Categories, represented by distinct colors, such as blue (like category 0, 8) predominantly occupied the left, while shades of red and orange (categories 1a and 3) veered towards the right. Even though a weak trend was discernible, the emergence of clearer clusters was a positive sign.



Lime explanations highlighted that a recurring challenge was the model's difficulty in accurately predicting category 6 for series_id = 1. However, an optimistic note was the model's improved precision in predicting series:4 as category 4, affirming its enhanced predictive capabilities.





Experiment B-4: Bidirectional GRU

The bidirectional GRU model incorporated an input layer, a bidirectional GRU layer, and an output layer with 10 units in the GRU layer. After a 30-epoch training regime, this model achieved training, validation, and test accuracies of 0.4796, 0.4482, and 0.4462, respectively. The training time stretched from 2 minutes 52 seconds to 4 minutes 27 seconds, yet the accuracy didn't depict considerable leaps. The results from confusion matrix and t-SNE plot closely resembled those of Experiment B-3. For a detailed breakdown, one can refer to the appendix titled Experiment B-4.

Experiment B-5: LSTM

The LSTM model was structured with an input layer, an LSTM layer, and an output layer, incorporating 10 units in the LSTM layer. Post a 30-epoch training, the model delivered training, validation, and test accuracies of 0.4421, 0.4457, and 0.4331, respectively, in a span of 2 minutes 9 seconds. The outcomes slightly underperformed compared to the GRU model (B-3). Again, the results mirrored the B-3 model in aspects like the confusion matrix and t-SNE plot. For in-depth insights, one can access the appendix under Experiment B-5.

Experiment B-6: Bidirectional LSTM

Our bidirectional LSTM network involved an input layer, a bidirectional LSTM layer, and an output layer. After training for 30 epochs, the model produced a training accuracy of 0.3864, a validation accuracy of 0.3708, and a test accuracy of 0.3683. The training duration expanded from 2 minutes 9 seconds to 4 minutes 15 seconds, yet no significant improvement in accuracy was observed, aligning with the performance of the unidirectional LSTM.

Insights from Experiment B

From our various experiments, the standout model in terms of performance was the GRU network (B-3). Our future endeavors will revolve around refining and optimizing this specific architecture. A recurrent observation across RNN types, be it the simple RNN, GRU, or LSTM, was that the bidirectional model didn't substantially overshadow its unidirectional counterpart. This can likely be attributed to the sequence length; while bidirectional models often shine in complex domains like natural language processing, simpler time-series problems might not leverage the model's full potential. In fact, these bidirectional models required extended training times and could be considered resource-intensive without commensurate performance gains.

Experiment C: Evaluating Impact of Node Count in GRU Layer

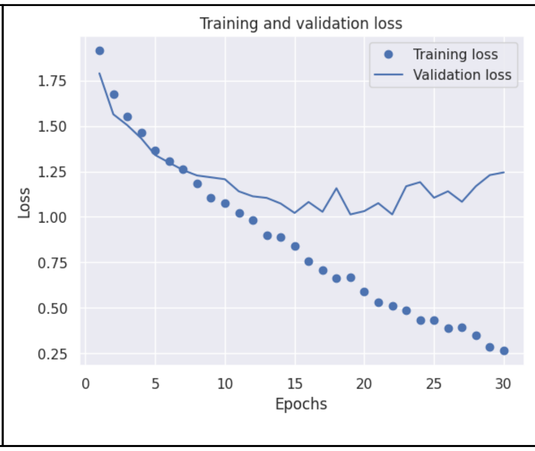
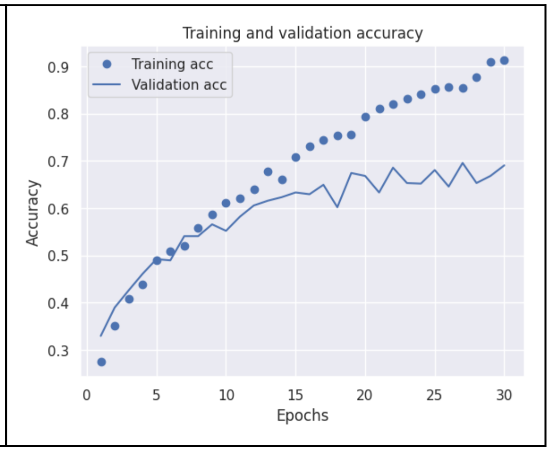
In our continued endeavor to refine and optimize our model architectures, Experiment C was initiated to assess the influence of varying node counts within the GRU layer on overall performance.

The model structure comprised an input layer, a GRU layer, and a concluding output layer. In this exploration, the GRU layer featured different node counts, specifically: 32, 64, 128, 256, and 512. As for the output layer, the Softmax activation function was utilized. The model was optimized using the Adam optimizer and trained against the "sparse_categorical_crossentropy" loss function. All other hyperparameters were maintained at their default values.

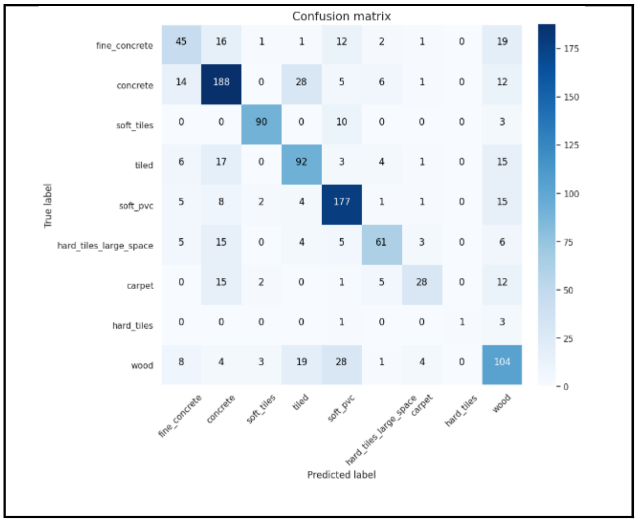
Upon training each model variant across 30 epochs, the standout performer was the model labeled C-5, which had 512 nodes in its GRU layer. This model achieved training, validation, and test accuracies of 0.9121, 0.6904, and 0.6877.

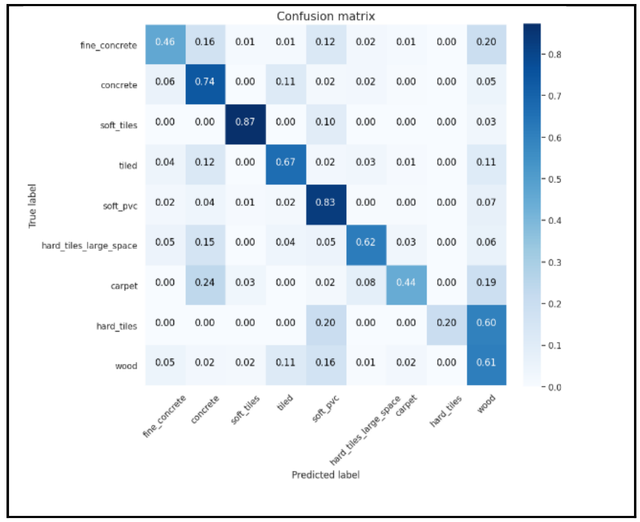
A striking observation from the experiment was the positive correlation between the number of nodes in the GRU layer and the resultant performance. Although training accuracy soared close to 90%, validation accuracy plateaued around the 60% mark.

Further insights were gleaned from dual plots generated using Matplotlib, juxtaposing training and validation loss (and likewise, accuracy) across epochs. A noticeable divergence between training and validation accuracies became evident from epoch 15 onward. This highlighted potential overfitting, suggesting the exploration of regularization techniques.

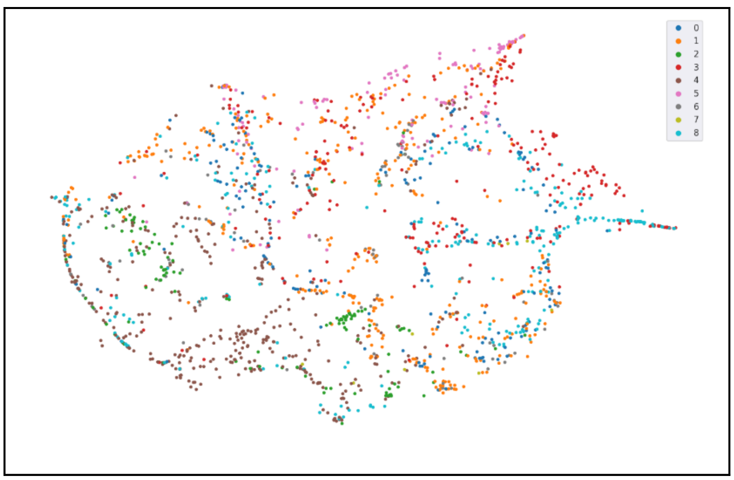
<table>
<tr>
<td>C-5</td>
<td>
</td>
<td>
</td>
</tr>
</table>

The confusion matrix, forming another part of our analysis. A perceptible need for further refinement. However, compared to model B-3, which was built with a mere 10 nodes in its GRU layer, a brighter illumination of diagonal cells in model C-5's matrix was an encouraging outcome.

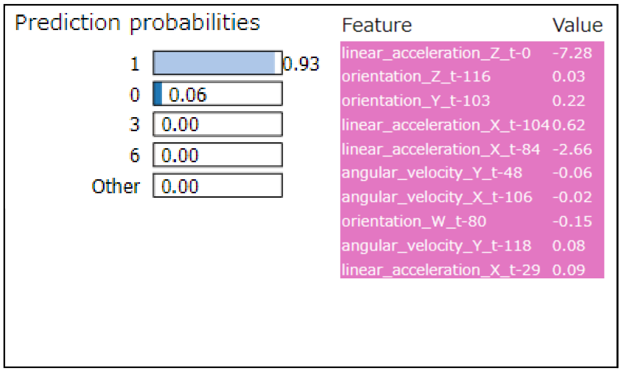


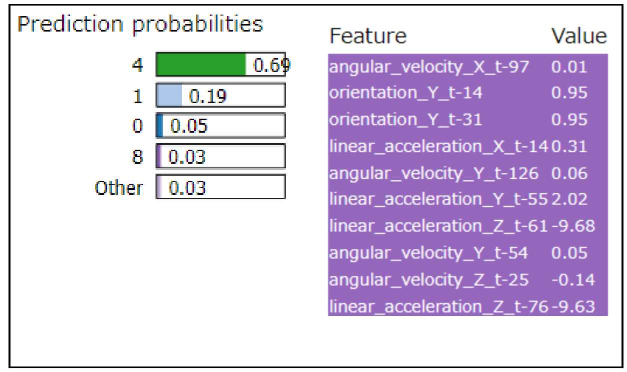


Additionally, a t-SNE plot of activation values was crafted. Distinct clustering trends in model C-5. For instance, a red cluster representing category:3 was prominently positioned in the upper right, whereas a green cluster, signifying category:2, anchored the lower left region.



Upon examining the Lime explanations, consistent challenges remained, particularly with the model's ability to predict category 6 for series_id = 1. Nevertheless, for series:4, model C-5 displayed heightened confidence in its prediction, further cementing its enhanced predictive prowess compared to prior models.





Detailed results of other variations explored within Experiment C are accessible in the appendix labeled "Experiment C."

Through Experiment C, it was ascertained that a GRU layer with a higher node count tends to yield superior performance. However, potential overfitting, as evidenced by the increasing divergence between training and validation accuracies, warrants further investigation. As a subsequent course of action, we intend to delve into regularization techniques to counteract this and further enhance the model's robustness.

Experiment D: Implementation of Early Stopping

To mitigate the issues of potential overfitting, as observed in our previous findings from Experiment C, and to optimize the utilization of our computational resources, Experiment D was structured around the incorporation of early stopping.

For this experiment, models established in Experiment C were used as base structures, upon which early stopping was applied. The methodology was designed to halt training once the model stopped improving, which, in essence, aimed to save computational time and resources.

Interestingly, upon application of early stopping, the model named D-4, equipped with 256 nodes in the GRU layer, emerged as the most superior. This model was trained over 23 epochs and achieved the following performance metrics:

- Training accuracy: 0.769
- Validation accuracy: 0.633
- Test accuracy: 0.6279

While it's notable that the test accuracy observed was marginally lower than that achieved in Experiment C. However, it significantly reduced training time, as evident in the accompanying table. Given our constraints related to computational power and time availability, the decision to integrate early stopping in our training process was reinforced.

Experiment E: Implementation of Dropout Regularization

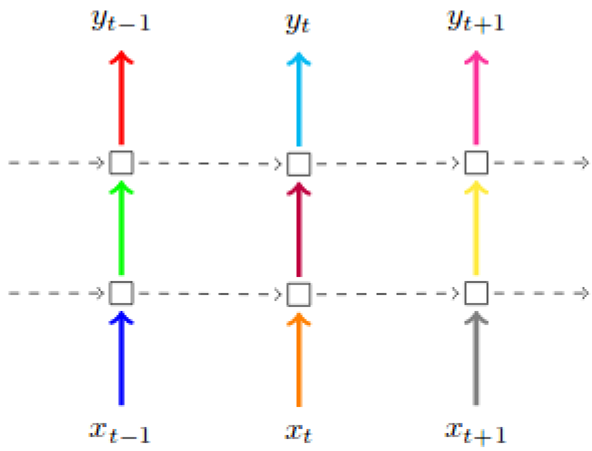
In Experiment E, we shifted our attention to dropout regularization, building upon the insights and the structure of model D-4 from the prior experiment. The goal was to counter overfitting and enhance the generalization capabilities of the model.

We designed GRU networks comprised of an input layer, a GRU layer, and an output layer. Dropout was incorporated as a regularization technique. Whereas the default fraction of units to drop for the linear transformation of the inputs is 0, in this experiment, we adjusted the dropout value ranging from 0.3 to 0.7. The output layer utilized the Softmax activation function, the optimizer was set to Adam, and the loss function chosen was "sparse_categorical_crossentropy". All other hyperparameter settings were left at their default.

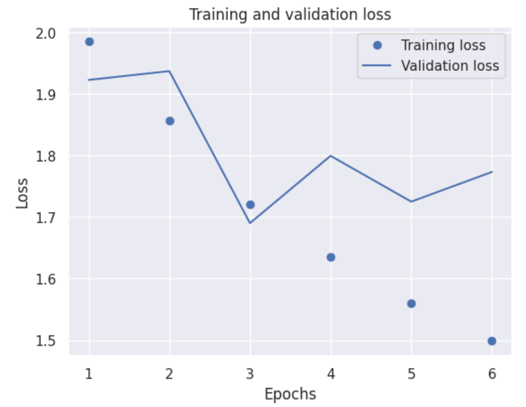
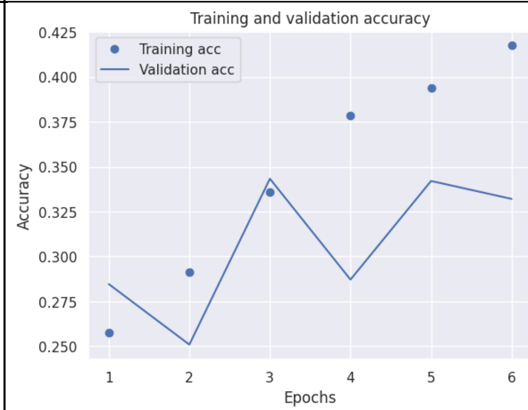
The most outstanding model of this experiment was E-1, which trained for 6 epochs. The performance metrics recorded for this model were:

- Training accuracy: 0.4175
- Validation accuracy: 0.3321
- Test accuracy: 0.3596

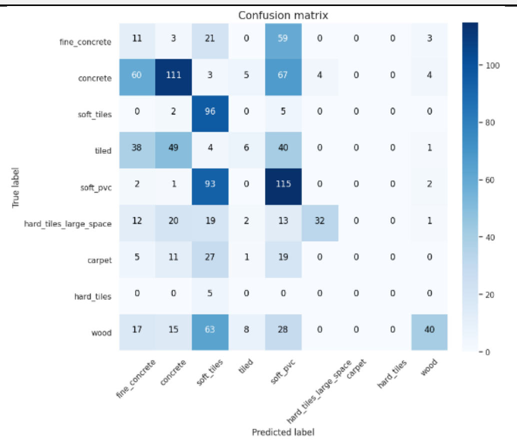
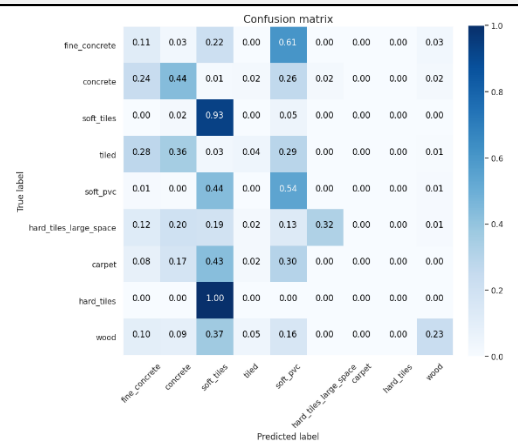
In the context of recurrent neural networks, dropout was introduced as a regularization strategy to the inputs, denoted by the vertical arrows leading from `x_t` to `y_t` (as proposed by Gal and Ghahramani in 2015). A recognized challenge with these models, as highlighted by Gal and Ghahramani, is their propensity to overfit. Dropout, when applied to recurrent layers, was found to be ineffective. Our experiment, regrettably, supports this claim, showcasing that standard dropout layers don't yield optimal performance.



Our utilization of Matplotlib for visualization indicated that both the validation loss and accuracy failed to stabilize, exhibiting oscillating patterns. As a result, the early stopping mechanism was triggered, causing premature cessation of training.

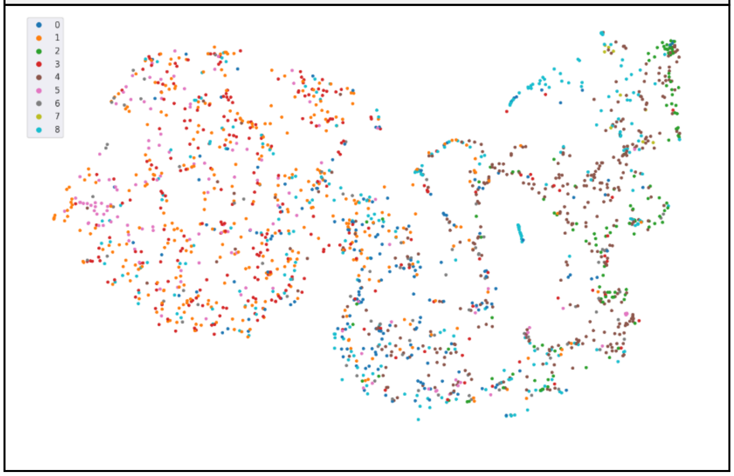
ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
E-1		

A glance at the presented confusion matrix revealed a decline in performance compared to model D-4. This was further confirmed by the lack of a pronounced diagonal, which is generally indicative of correct predictions.

The heat map of the confusion matrix	The heat map of the normalized confusion matrix
	

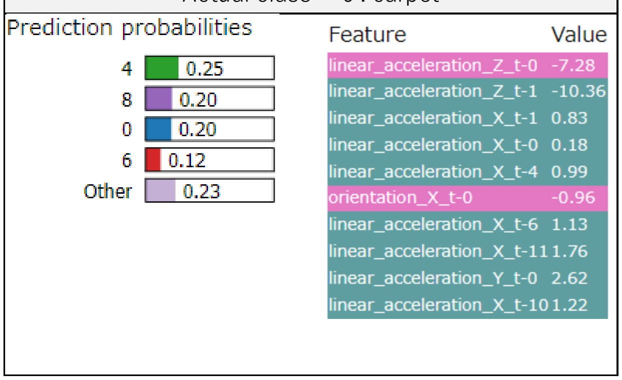
Further, the t-SNE plot of activation values showcased some discernible patterns. Categories like 0 and 8 (denoted in blue) were predominantly on the right, while categories such as 1a and 3 (represented in red and orange) were on the left. However, the trends appeared weakened and more interspersed than before.

t-SNE plot

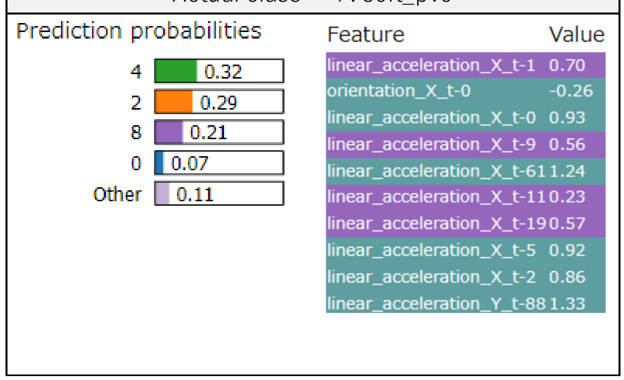


Lastly, our Lime explanation divulged persistent challenges in model predictions. For instance, the model consistently misjudged series_id = 1 as category 6. Furthermore, for series:4, its confidence in predicting it as category 4 waned, with the model frequently misclassifying it as category 2.

Lime explanation (prediction for series_id = 1)
Actual class = 6 : carpet



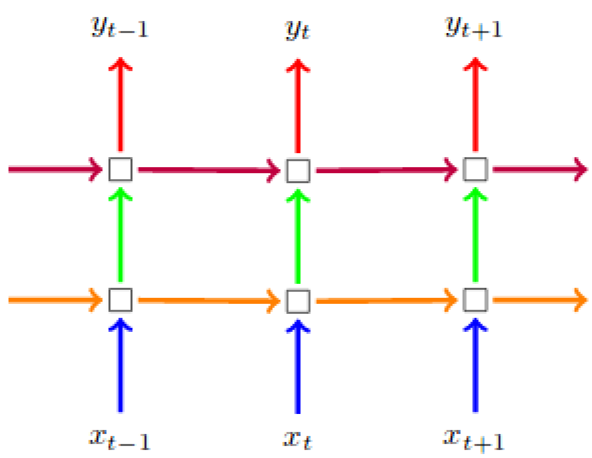
Lime explanation (prediction for series_id = 4)
Actual class = 4 : soft_pvc



Experiment E highlighted the challenges of utilizing dropout regularization within recurrent neural networks, particularly GRU layers. The performance metrics underscored the model's struggle to generalize effectively. Future work may need to explore alternative regularization techniques or revisit the architecture to ensure a better balance between model complexity and generalization.

Experiment F: Implementing Recurrent Dropout

In Experiment F, we ventured into the territory of recurrent dropout, drawing inspiration from Gal and Ghahramani's 2015 study. In their research, they introduced the concept of recurrent dropout, which is strategically applied to "drop" or mask connections between recurrent units. For illustrative purposes, these connections would correspond to the horizontal arrows in the diagrams that we reference.



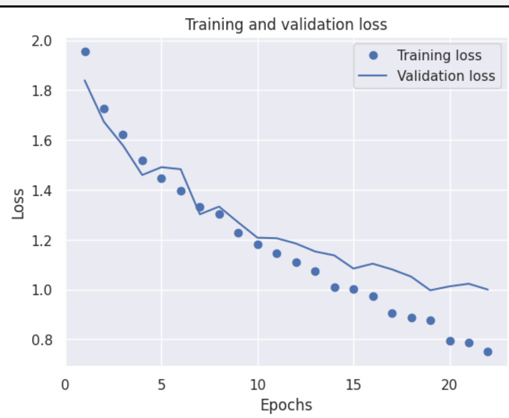
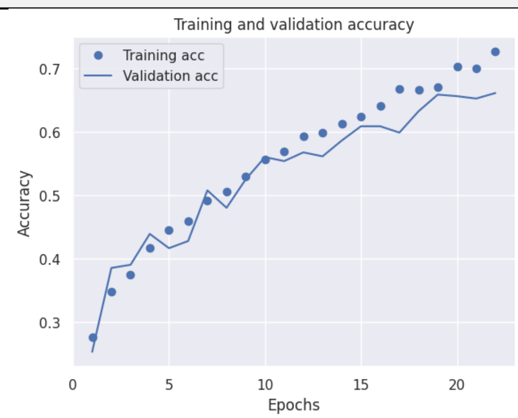
Leveraging the architecture of model D-4 as a foundation, we incorporated recurrent dropout into our GRU network, which consists of an input layer, a GRU layer, and an output layer. The dropout, in this context, pertains to the fraction of units to be dropped during the linear transformation of the recurrent state. While the default is set to 0, for this experiment, we varied the dropout values from 0.3 to 0.7. The output layer featured the Softmax activation function, while the optimizer and loss function remained as Adam and "sparse_categorical_crossentropy", respectively. All other hyperparameters were kept at default settings.

Of all the iterations, model F-1 emerged as the top performer, completing training over 22 epochs. The key performance metrics for this model were:

- Training accuracy: 0.7267
- Validation accuracy: 0.6617
- Test accuracy: 0.6535

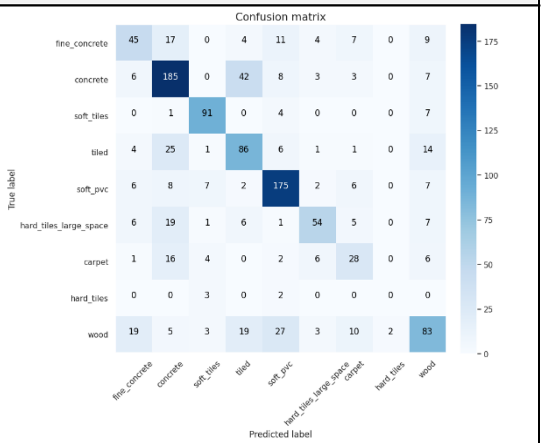
It's worth noting that model F-1 demonstrated superior performance compared to model D-4. However, models F-2 and F-3, with recurrent dropout values set to 0.5 and 0.7, lagged behind model F-1. This could imply an excessive loss of recurrent information in these models, which negatively impacted their effectiveness.

Our visual analysis using Matplotlib showed promising signs. The training and validation curves were closely aligned, lacking the pronounced gap observed in Experiment C. This suggests that model F-1 might be a more generalized representation.

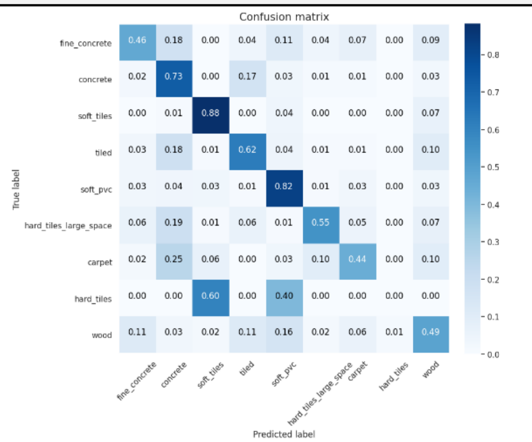
ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
F-1		

The confusion matrix revealed continued challenges. One noticeable pattern was the model's tendency to misclassify "hard_tile" as "soft_tiles". Nonetheless, there was a perceptible increase in the correctly predicted instances, evidenced by a more pronounced diagonal in the matrix.

The heat map of the confusion matrix

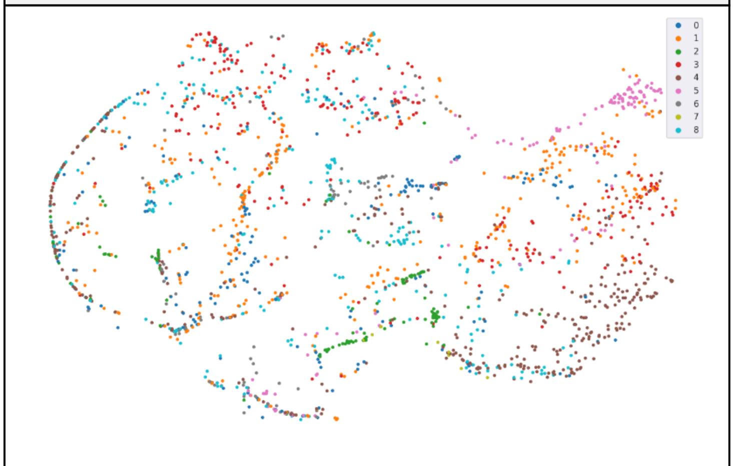


The heat map of the normalized confusion matrix



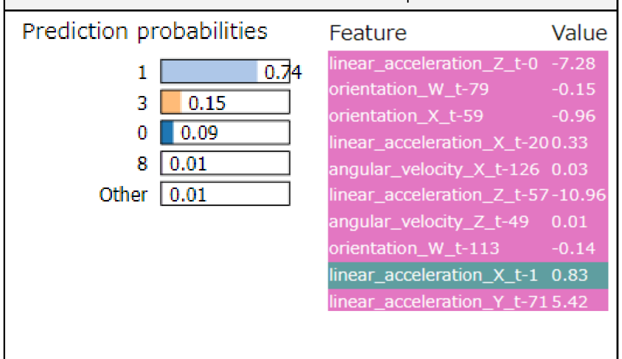
The t-SNE plots of activation values were also assessed. Some trends were still discernible, for example, categories such as 0 and 8 (colored in blue) predominantly featured on the right, whereas categories 1a and 3 (in hues of red and orange) appeared on the left. While there were observable clusters, they appeared to be more mixed and less distinct than in previous experiments.

t-SNE plot

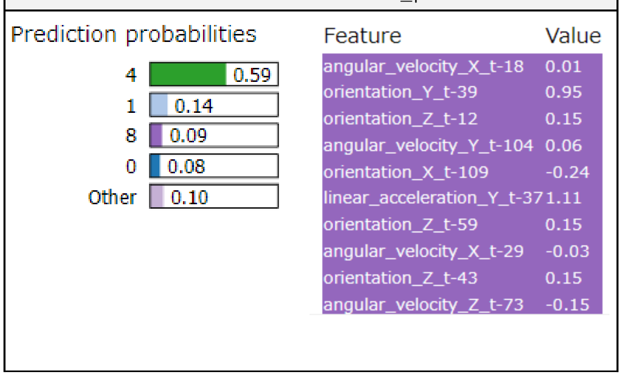


Lastly, the Lime explanations painted a more positive picture for this experiment. Model F-1 continued to mispredict series_id = 1 as category 6. However, when it came to series:4, it demonstrated a clear and confident prediction of it as category 4, which is a marked improvement over the results of model E-1, which employed traditional dropout.

Lime explanation (prediction for series_id = 1)
Actual class = 6 : carpet



Lime explanation (prediction for series_id = 4)
Actual class = 4 : soft_pvc



Experiment F underscored the potential benefits of employing recurrent dropout in GRU models, a finding that resonates with Gal and Ghahramani's 2015 study. The experiment exhibited that striking the right balance in dropout values is crucial to harnessing the benefits of this technique. Future endeavors might delve deeper into fine-tuning these values and integrating other optimization techniques to further elevate the model's accuracy and generalization capabilities.

Experiment G: Implementation of Weight Decay (L1, L2, L1-L2)

In the pursuit of refining our models further, Experiment G centered on the application of weight decay regularization. Using the architecture of the promising model F-1 as a template, we set out to implement weight decay techniques such as L1, L2, and their combined form, L1-L2. This GRU network still maintained its foundational structure, consisting of an input layer, a GRU layer, and an output layer. The final output layer employed the Softmax activation function, with the optimizer being Adam and the loss function specified as "sparse_categorical_crossentropy". We stuck to the default settings for other hyperparameters.

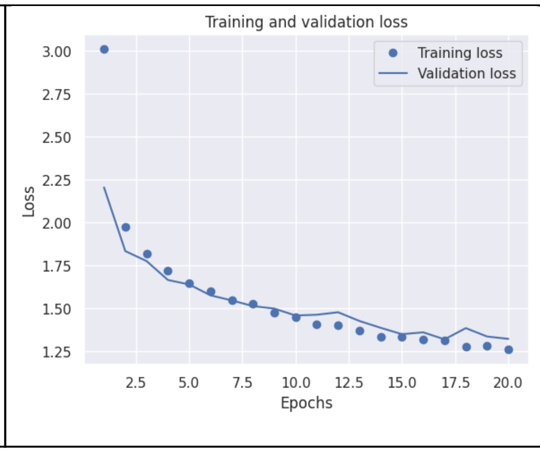
Out of the iterations, model G-2 stood out, completing its training over a span of 30 epochs. The performance metrics for this model were:

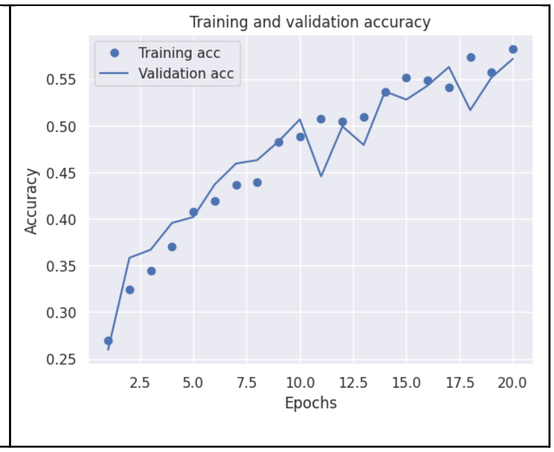
- Training accuracy: 0.582
- Validation accuracy: 0.5718
- Test accuracy: 0.5599

However, a comparison with model F-1 revealed a decline in performance for model G-2. Weight decay regularization, while controlling the magnitude of the weights, seemed to have inadvertently minimized the importance of some critical weights in the model, potentially obscuring pivotal information needed for predictions.

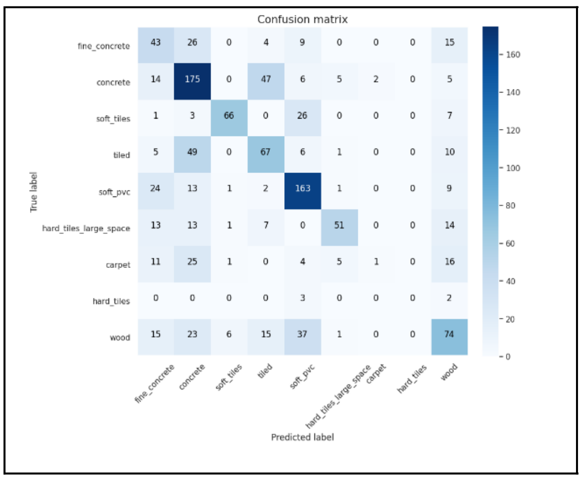
Our visual analysis, courtesy of Matplotlib, revealed a few insights. Unlike Experiment C, the training and validation curves for model G-2 were in close alignment. However, the speed at which the loss was reduced during the training was noticeably slower.

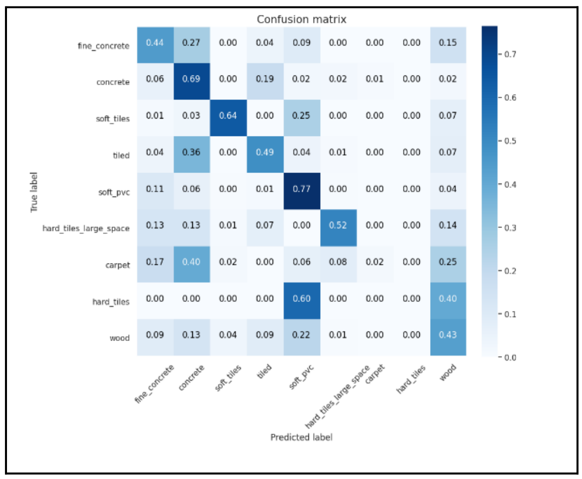
G-2



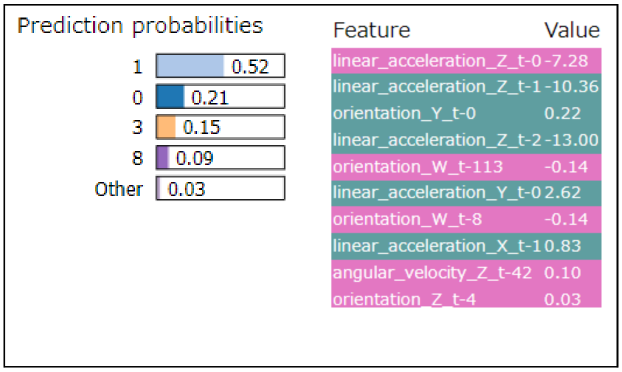


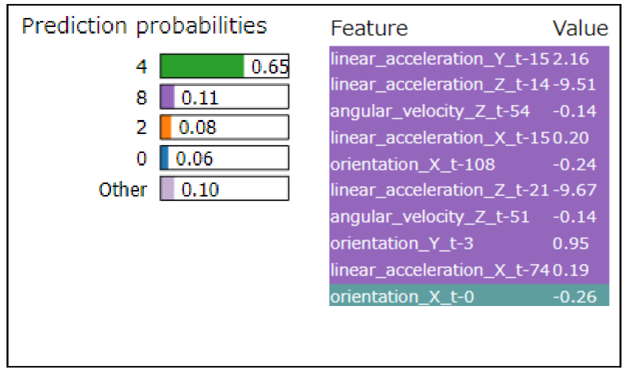
A deeper dive into the confusion matrix exposed further classification errors. Most notably, there was a recurrent misclassification of the category "carpet" as "concrete".



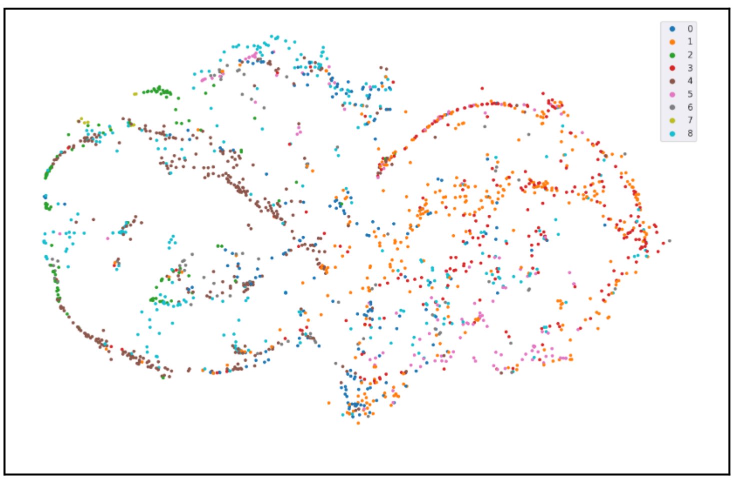


The Lime explanation corroborated this observation. The model consistently struggled with predicting the "carpet" category (6) for series_id = 1. The confusion matrix provided clarity on this front, revealing that the model was consistently classifying the "carpet" as "fine concrete" (1).





In our t-SNE visualization of activation values, we could still discern some patterns. Categories such as 0 and 8, colored in blue, appeared predominantly on the left side. In contrast, categories 1a and 3, rendered in shades of red and orange, marked their presence on the right. Nevertheless, the clusters were more dispersed and the trends less pronounced than in our previous experiments.



Experiment G underlined the complexities associated with introducing weight decay in models. While the intention was to strike a balance and avoid overfitting, the application of L1, L2, and their combined form led to a suboptimal model that often misclassified categories. This experiment reinforces the notion that while regularization techniques are essential tools in a researcher's toolkit, their indiscreet application can sometimes lead to unintended consequences.

Experiment H: Incorporation of Additional Hidden Layers (2 GRU layers)

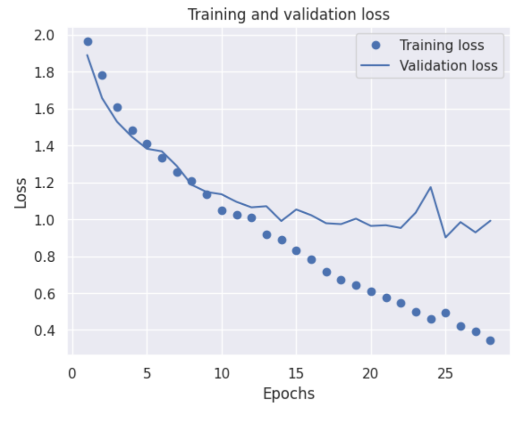
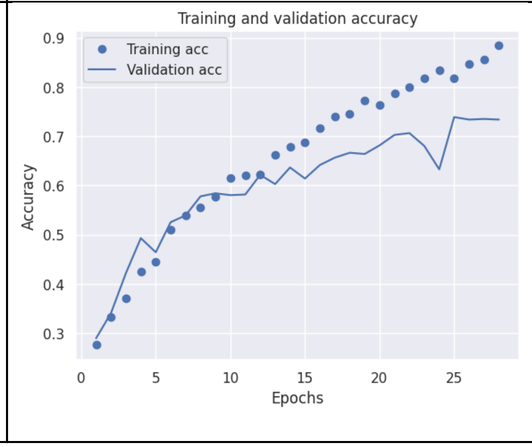
In an endeavor to harness the prowess of deeper neural networks, Experiment H involved the adaptation of model F-1. This new iteration featured a GRU network with an input layer, a structure comprising two GRU layers, followed by the customary output layer. The output layer once again was designed using the Softmax activation function. For the optimization of the model, Adam was employed, while "sparse_categorical_crossentropy" was used as the loss function. Apart from these specifications, other hyperparameters were maintained at their default settings.

The resulting Model H-1 underwent training for a total of 28 epochs. The performance statistics revealed:

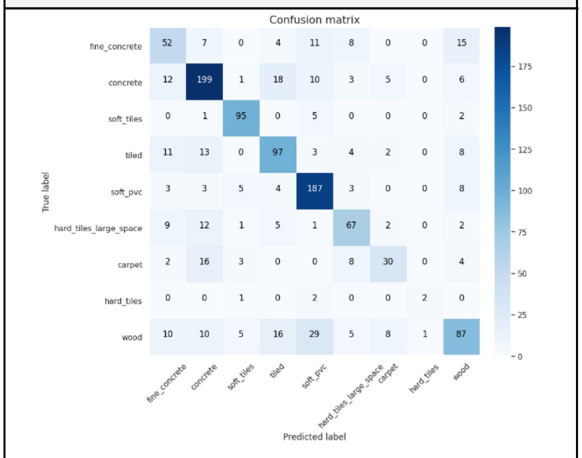
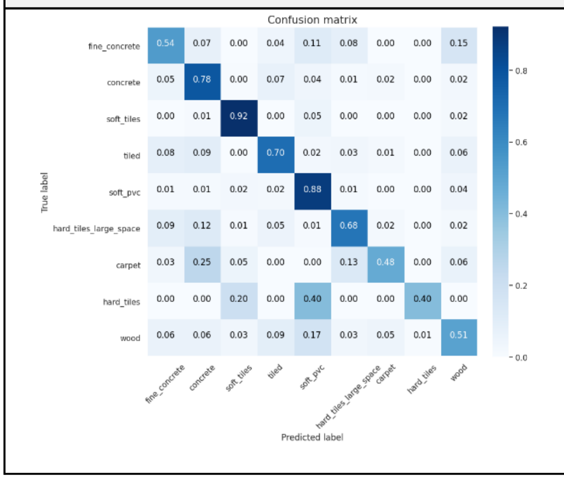
- Training accuracy: 0.8842
- Validation accuracy: 0.7341
- Test accuracy: 0.7139

This demonstrates that Model H-1 reached the best score in terms of test accuracy. Goodfellow, Bengio, and Courville (2016) had alluded to this kind of outcome, stating that deeper networks, courtesy of additional layers, augment the model's capacity to understand intricate relationships embedded within the data.

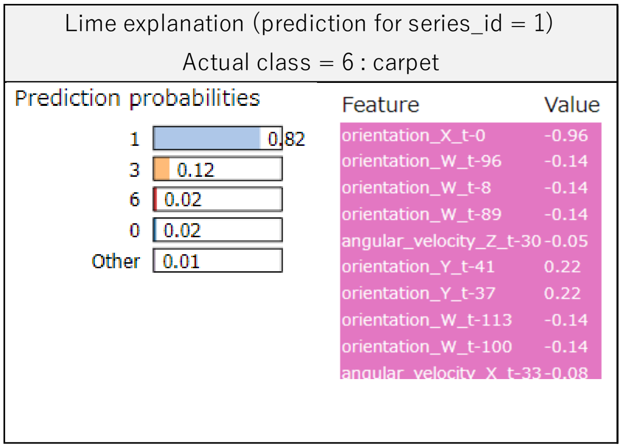
Visual aids created through Matplotlib showcased the trajectory of training and validation loss (and accuracy) side by side across the epochs. However, a slight deviation between the training and validation curves was observed, hinting towards potential overfitting.

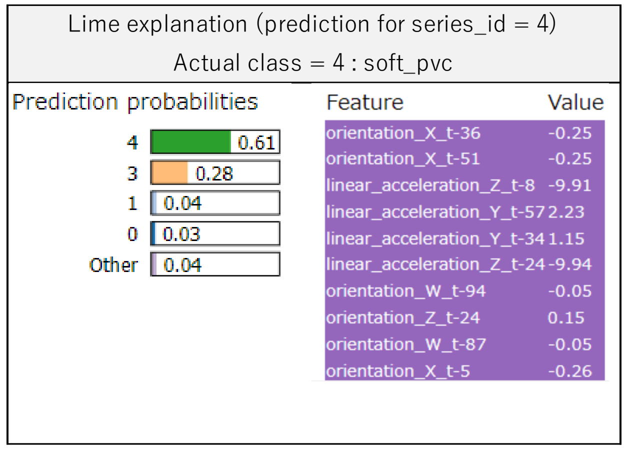
ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
H-1		

When assessing the confusion matrix for Model H-1, an evident improvement was discernible in comparison to preceding experiments. The matrix presented a predominant diagonal trend, which is often an indicator of a model's adept classification ability. However, some old challenges persisted. One prominent misclassification was that of the "carpet" category, which was frequently mistaken for "concrete".

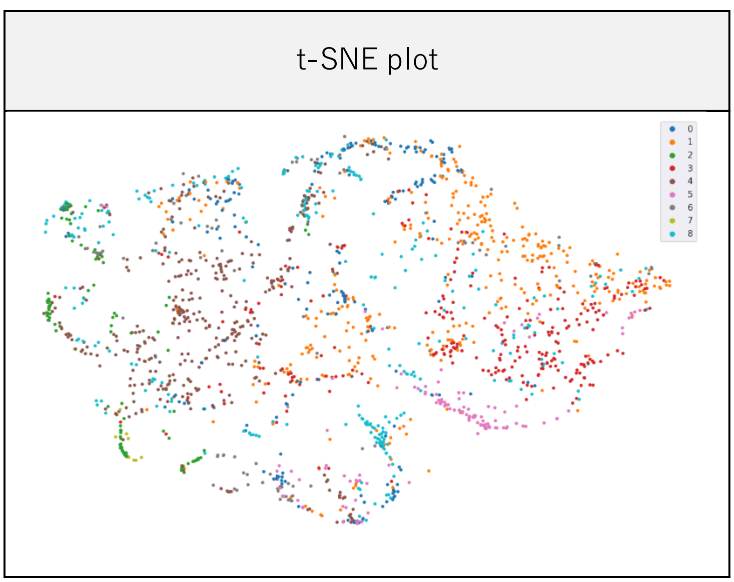
The heat map of the confusion matrix	The heat map of the normalized confusion matrix
	

Delving into the Lime explanations, Model H-1 still exhibited issues in classifying category 6: "carpet" for series_id = 1. The model's propensity to mislabel it as category 1: "fine concrete" was undeterred. The Lime analysis pinpointed "orientation_x_t-0" as a dominant factor influencing this specific prediction. Conversely, when classifying series_id=4 (actual class: "soft_pvc"), the model displayed remarkable accuracy. This accurate classification was a cumulative effect of linear acceleration data, complemented by orientation information.





Lastly, the t-SNE visualization of activation values still projected discernible trends. Categories denoted in blue hues (such as 0 and 8) were predominantly plotted on the left side, whereas those in red and orange shades (like 1a and 3) were clustered on the right. An enhancement from previous experiments was the heightened clarity and distinctiveness of the clusters in this visualization.



Experiment H emphasized the power and potential of deeper neural networks. With the introduction of an additional GRU layer, the model's capacity to comprehend and map the intricate interplay of features in the data amplified. This resulted in superior accuracy and clearer visualizations. However, specific recurrent challenges, like the misclassification of the "carpet" category, indicate the possibility of further refinements and experiments in the future.

Experiment I: Integration of Additional Hidden Layers (1d CNN + GRU layer)

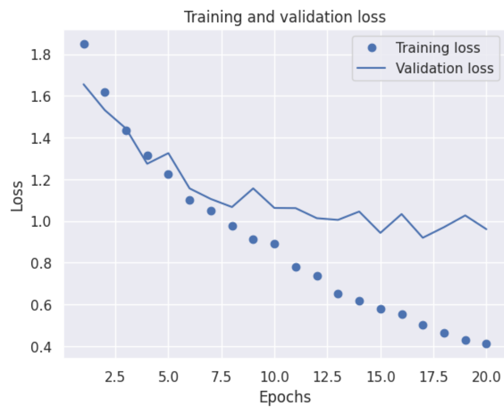
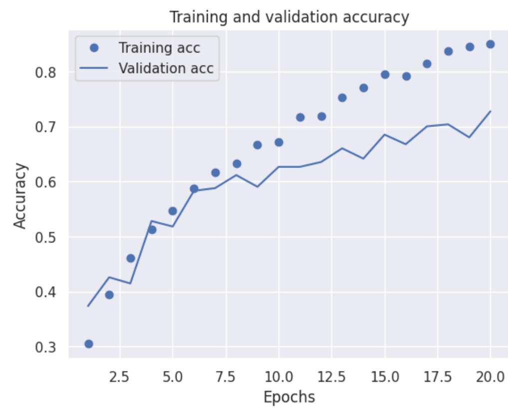
To further explore the capabilities of hybrid neural network architectures, Experiment I was executed. It involved making alterations to the structure of model F-1, where a 1-dimensional convolutional layer (1d CNN) was integrated before the GRU layer. The final architecture comprised of the 1dCNN layer, a GRU layer, and a concluding output layer equipped with the Softmax activation function. Adam was the chosen optimizer, and "sparse_categorical_crossentropy" remained as the loss function. All other hyperparameters were set to their default values.

Post the training cycle of 20 epochs, Model I-1 yielded the following performance metrics:

- Training accuracy: 0.8499
- Validation accuracy: 0.7278
- Test accuracy: 0.6728

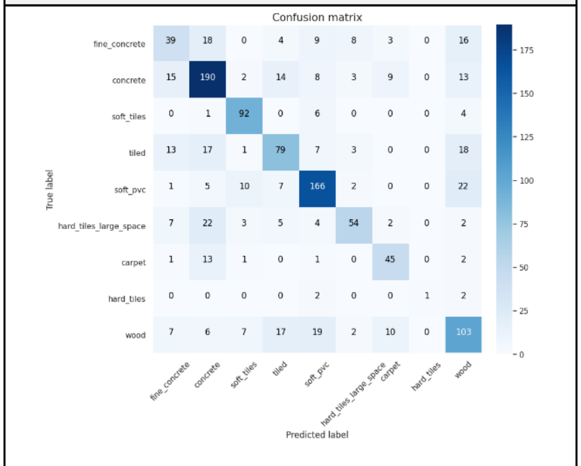
Model I-1 stands out as one of the top-performing models across our suite of experiments.

Matplotlib was once again employed to craft visual representations, plotting training and validation losses (and accuracies) across the epochs. A noticeable divergence between the training and validation plots was observed. The level of overfitting was discernibly higher in Model I-1 compared to Model H-1, which had two GRU layers. This difference, perhaps, hints at Model H-1 being more adept at generalizing across unseen data.

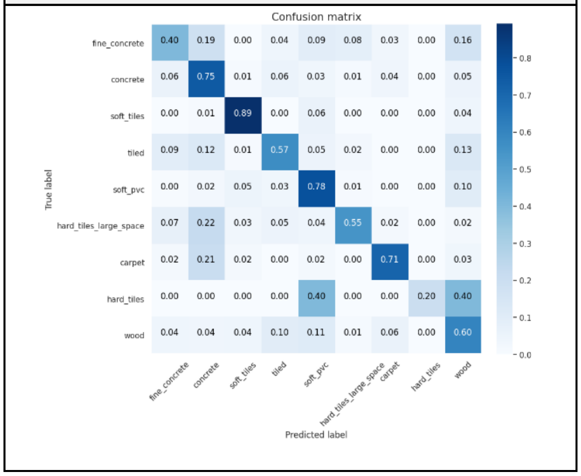
ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
I-1		

The confusion matrix for Model I-1 revealed persistent challenges in classifying "hard_tiles." Across all models in our experiments, an evident difficulty in accurately predicting the "hard_tiles" category was a constant. A plausible explanation for this recurring misclassification could be the limited representation of "hard_tiles" within the dataset. To alleviate this challenge, sourcing additional data pertaining to "hard_tiles" might be a potential solution.

The heat map of the confusion matrix

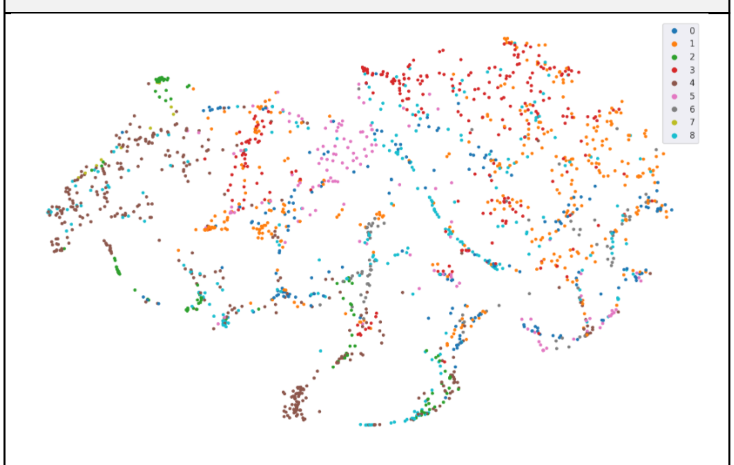


The heat map of the normalized confusion matrix



Lastly, t-SNE visualization was conducted for Model I-1's activation values. Trends in the plot mirrored some from previous experiments. Categories portrayed in blue hues (like 0 and 8) predominantly occupied the left side, while categories in shades of red and orange (like 1a and 3) were predominantly seen on the right. However, an improvement from prior models was evident – the clusters in this visualization were more distinct and cleanly separated.

t-SNE plot



The introduction of a hybrid neural architecture in Experiment I, consisting of a 1d CNN layer followed by a GRU layer, has shown promising results. Though it demonstrated a strong ability to model and predict various categories, overfitting remains a concern. The persistent challenge of "hard_tiles" classification across all models points to a potential need for better data representation.

Experiment J: Incorporation of Additional Hidden Layers (3 GRU layers)

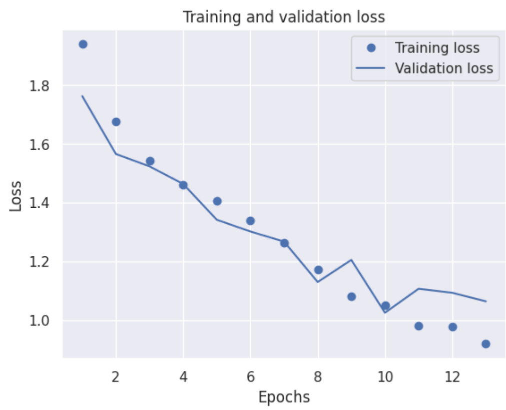
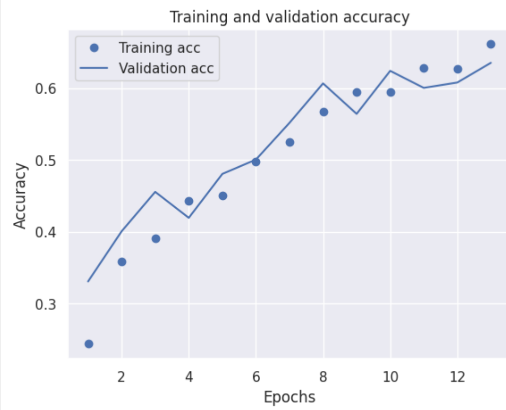
In the realm of deep learning, the potential benefits of stacking more layers often spark curiosity. Experiment J, thereby, delves into this avenue. Stemming from the architecture of model H-1, the neural network under this experiment was structured to house three GRU layers, placed between the input and output layers. The output layer was armed with the Softmax activation function. Adam was appointed as the optimizer and "sparse_categorical_crossentropy" as the loss function, with other hyperparameters retained in their default configuration.

Model J-1 underwent a training session lasting 13 epochs, at the culmination of which the following metrics were documented:

- Training accuracy: 0.6613
- Validation accuracy: 0.6355
- Test accuracy: 0.6098

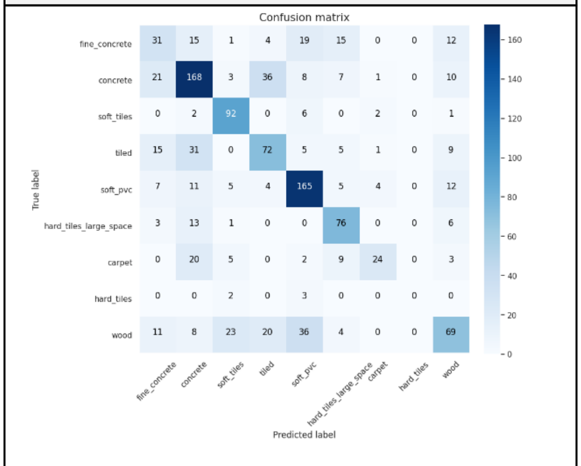
Interestingly, Model J-1 reported a decline in test accuracy in contrast to its predecessor, Model I-1. A sentiment mirrored in the works of Goodfellow, Bengio, and Courville (2016) suggests that stacking layers can aid in modeling intricate relationships inherent within data. Paradoxically, the introduction of the third GRU layer in our architecture didn't render favorable outcomes. A plausible explanation for this could be the infamous vanishing gradient problem, which becomes more probable with increasing layers, especially in RNNs.

A visual representation was created using Matplotlib, showcasing the training and validation losses and accuracies against the epochs. An erratic trend was noticed in the validation loss and accuracy plots, characterized by inconsistencies and oscillations.

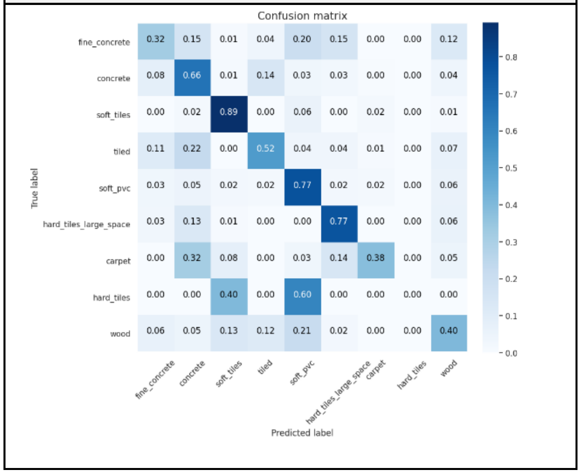
ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
J-1		

The confusion matrix for Model J-1 further elucidated the decline in its predictive prowess. The matrix appeared increasingly dispersed, marked by a notable prevalence of off-diagonal cells, indicating misclassifications.

The heat map of the confusion matrix

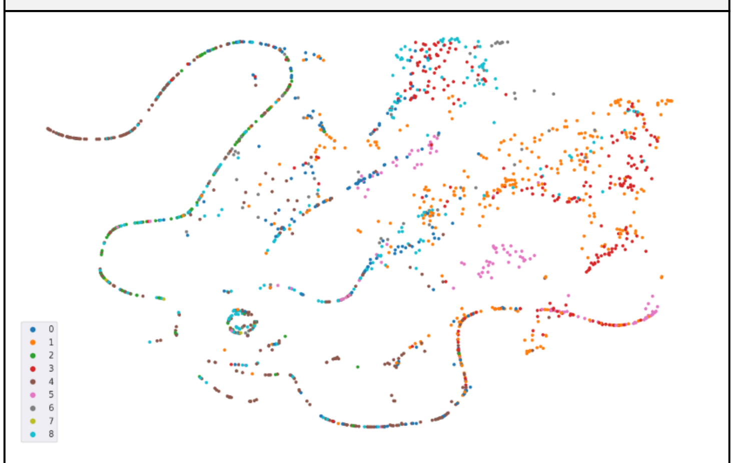


The heat map of the normalized confusion matrix



Subsequent to this, a t-SNE visualization was generated for the activation values of Model J-1. Although some patterns could still be discerned, this visualization manifested in a somewhat snake-like configuration. Broadly, categories painted in blue hues (like 0 and 8) were discernible on the left, while the shades of red and orange (representing categories like 1a and 3) populated the right. However, unlike prior experiments, clearer and more defined clusters were visible.

t-SNE plot



Experiment J offers an intriguing insight into the implications of excessive layering in recurrent networks. Although deep architectures are often celebrated for their capacity to comprehend complex relationships, they are not without challenges. The vanishing gradient problem, inherent to deep RNNs, might be the culprit behind the sub-optimal performance of Model J-1.

Conclusion

The advent of autonomous vehicles is undeniably groundbreaking. Yet, the road ahead for these futuristic vehicles is riddled with challenges. Foremost among these challenges is their ability to adeptly recognize and adapt to various road surfaces, especially under adverse weather conditions. Errors in this domain could lead to devastating consequences, imperiling the very lives depending on them.

This research ventured into the domain of recurrent neural networks (RNN) and the Long Short-Term Memory (LSTM) model, aiming to harness their prowess in predicting road surface types. Utilizing the extensive IMU sensor data from the "CareerCon 2019 - Help Navigate Robots" dataset, the primary objective was to accurately discern between nine distinct floor types. Beyond the theoretical, the research aimed to achieve a practical model with an ambitious accuracy benchmark of 97.53%, thus contributing to safer and more reliable robot navigation.

Our best model is model H-1(architecture shown below) achieved 71.39% in test accuracy.

Across a series of meticulous experiments, ranging from the comparative analysis of various RNN types to node count evaluations and complex hybrid models, several insights were gleaned:

1. Performance Nuances: The GRU network emerged as a frontrunner in our investigations. Yet, the pursuit of perfection remains, as even this standout model requires further refinement. Additionally, while bidirectional models are often heralded for their prowess, their performance didn't significantly overshadow unidirectional models in our context.

2. Tackling Overfitting: As node count increased, performance enhancement was observed, albeit with potential overfitting. Future directions should involve integrating regularization techniques to manage this.

3. Balancing Model Complexity and Generalization: Throughout our experiments, particularly in those involving dropout regularization and weight decay, the delicate equilibrium between complexity and generalization became evident. The indiscriminate implementation of regularization techniques sometimes yielded adverse effects.

4. Deep Architectures: While Experiment H showcased the potential of deeper networks, Experiment J served as a testament to the challenges inherent in overly deep RNNs, particularly the vanishing gradient problem.

In essence, the voyage into the world of autonomous vehicle safety through neural networks has been enlightening. While milestones have been achieved, the journey is far from over. As we stand at this juncture, the recommendations are clear: Pursue further model refinement, explore innovative regularization techniques, and always maintain a keen focus on achieving the perfect balance between model complexity and real-world applicability. The future of transportation safety hinges on these very advancements.

.

References

Fortune Business Insight. 2020. "Facial Recognition Market," July. https://www.fortunebusinessinsights.com/industry-reports/facial-recognition-market-101061

McCulloch, Warren S, and Walter Pitts. 1943. “A Logical Calculus of the Ideas Immanent in Nervous Activity.” _Bulletin of Mathematical Biology_ 5, no. 4 : 115–133.

Alzubaidi, Laith, Jinglan Zhan, Amjad J Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J Santamaría, Mohammed A Fadhel, Muthana Al-Amidie, and Laith Farhan. 2021. “Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions.” _Journal of Big Data_ 8, no. 53.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. “Deep Learning.” _Nature_ 521: 436–444. https://www.nature.com/articles/nature14539.

LeCun, Yann, Yoshua Bengio, Leon Bottou, and Patrick Haner. 1998. “Gradient-Based Learning Applied to Document Recognition.” _Proceedings of the IEEE_ 86, no. 11: 2278 – 2324.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. & Polosukhin, I. (2017). "Attention is all you need." _Advances in Neural Information Processing Systems_ (p./pp. 5998--6008), . https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Radford, A., & Narasimhan, K. (2018)." Improving Language Understanding by Generative Pre-Training." https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf

Schuster, Mike and Kuldip K. Paliwal, 1997. "Bidirectional recurrent neural networks," in _IEEE Transactions on Signal Processing_, vol. 45, no. 11, pp. 2673-2681, Nov. 1997, doi: 10.1109/78.650093.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. 2016. "Deep Learning". The MIT Press.

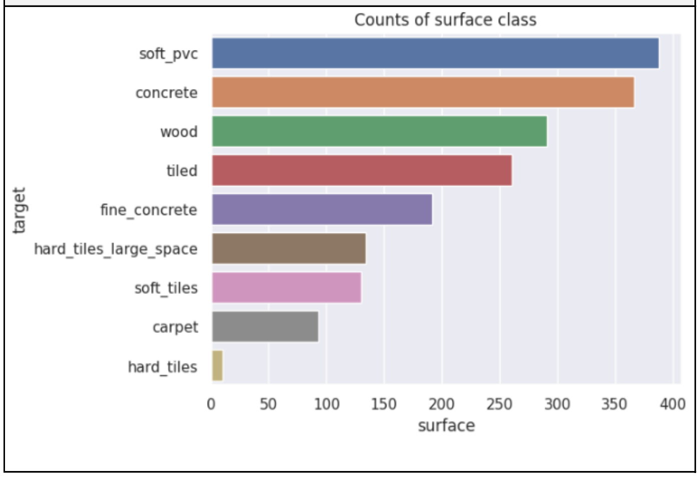
Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. 2016. "'Why Should I Trust You?': Explaining the Predictions of Any Classifier." arxiv, February 16, 2016. https://arxiv.org/abs/1602.04938.

Gal, Yarin, and Zoubin Ghahramani. 2015. "A Theoretically Grounded Application of Dropout in Recurrent Neural Networks." arxiv, December 16, 2015. https://arxiv.org/abs/1512.05287.

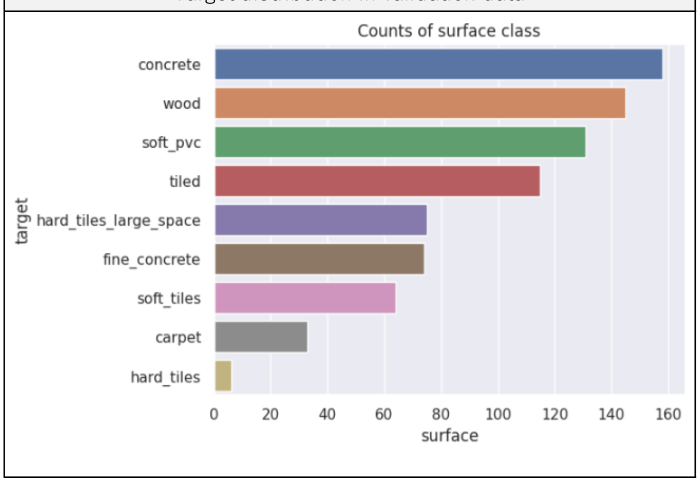
Appendix

Category distribution in train, validation, test dataset

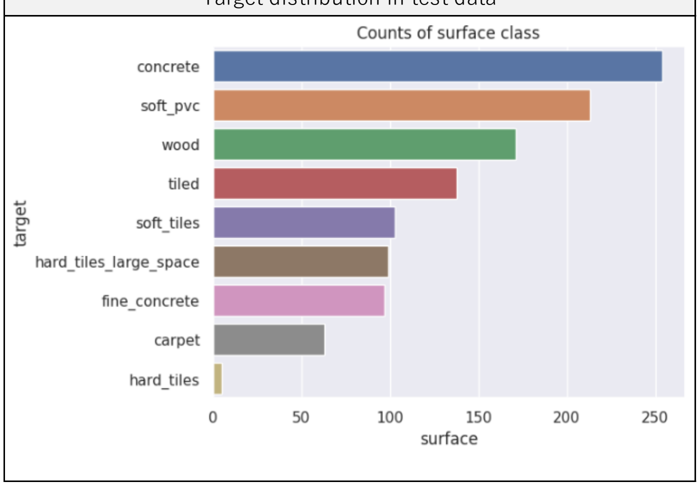
Target distribution in train data



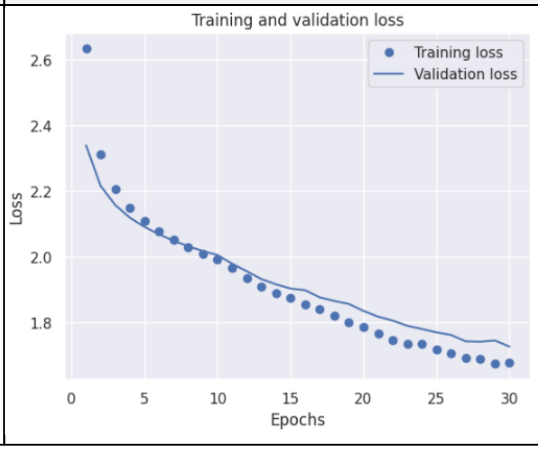
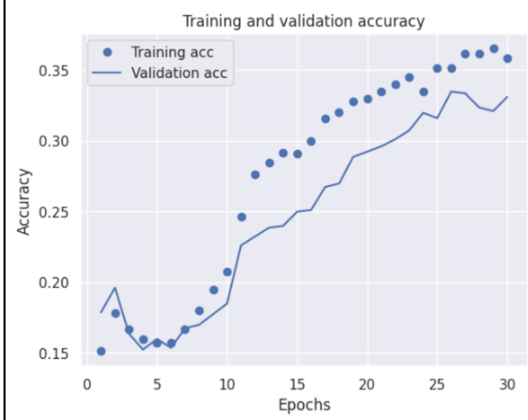
Target distribution in validation data



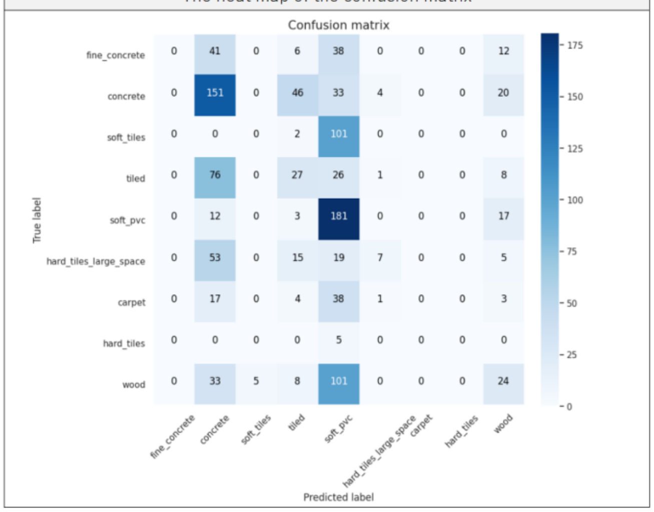
Target distribution in test data



Experiment B-1

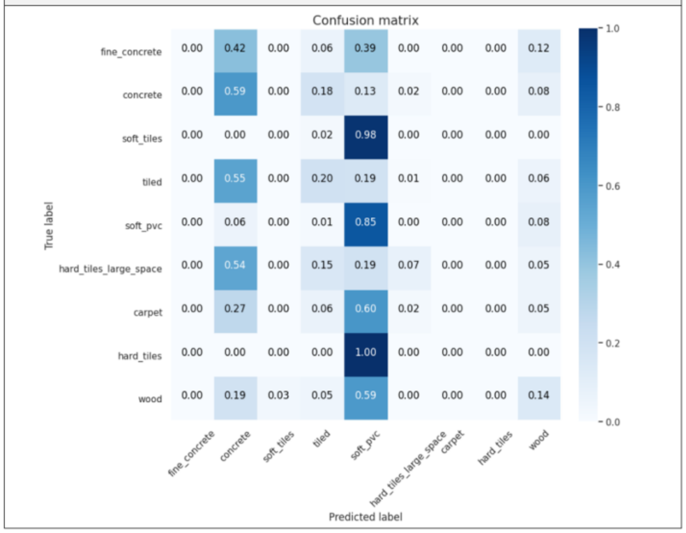
ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
B-1		

The heat map of the confusion matrix

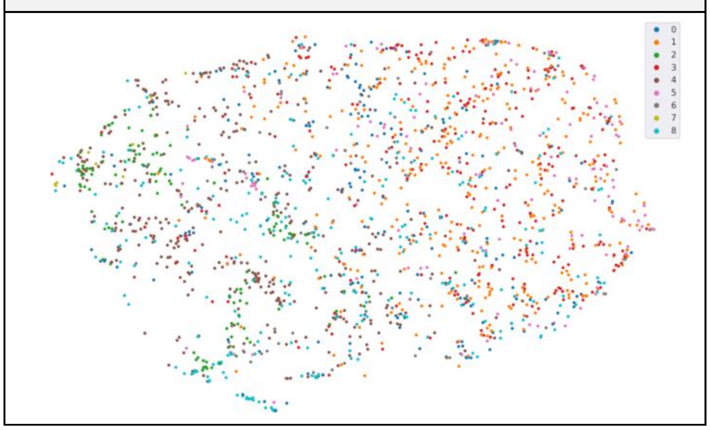


Sensor Data Analysis

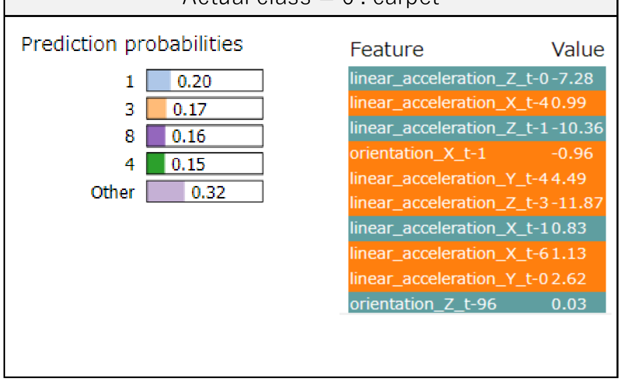
The heat map of the normalized confusion matrix



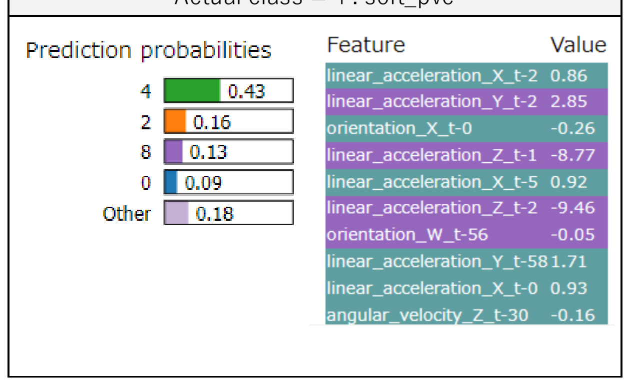
t-SNE plot



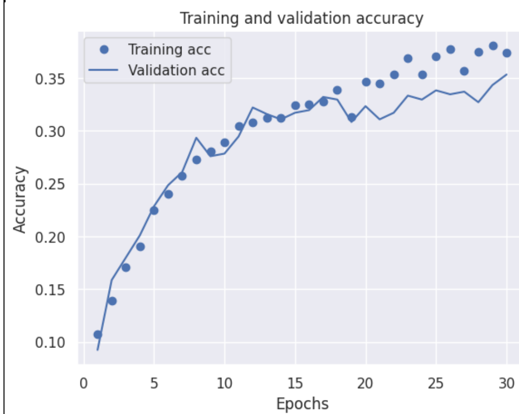
Lime explanation (prediction for series_id = 1)
Actual class = 6 : carpet



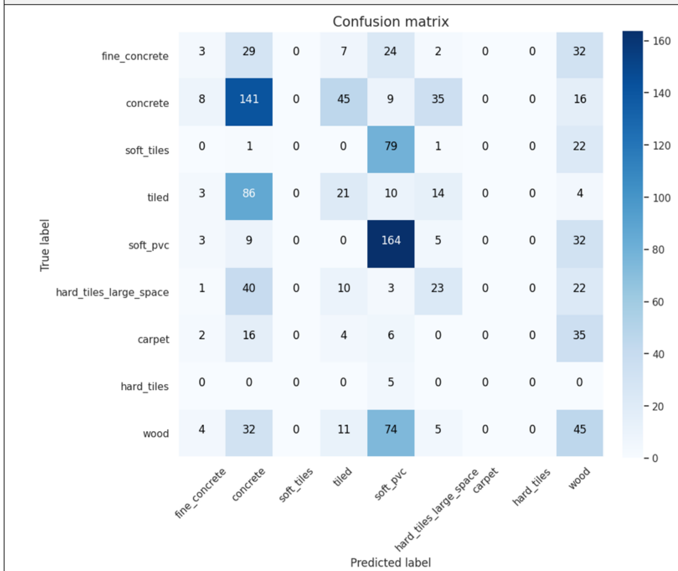
Lime explanation (prediction for series_id = 4)
Actual class = 4 : soft_pvc



Experiment B-2

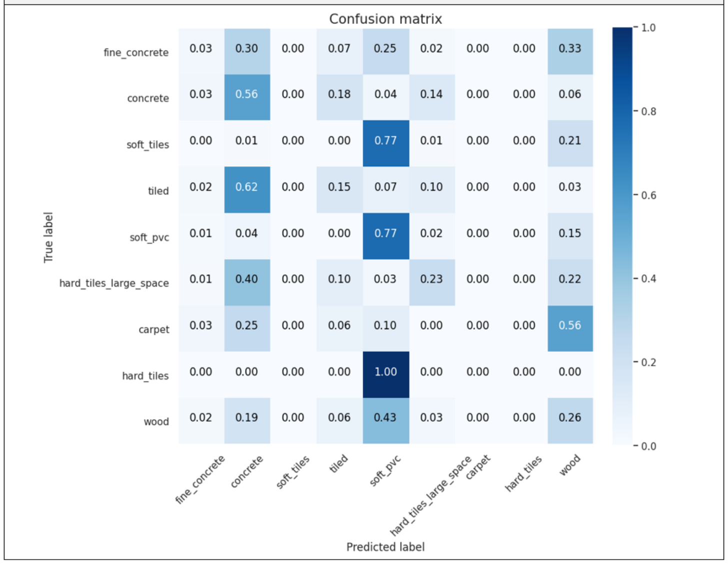
ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
B-2		

The heat map of the confusion matrix

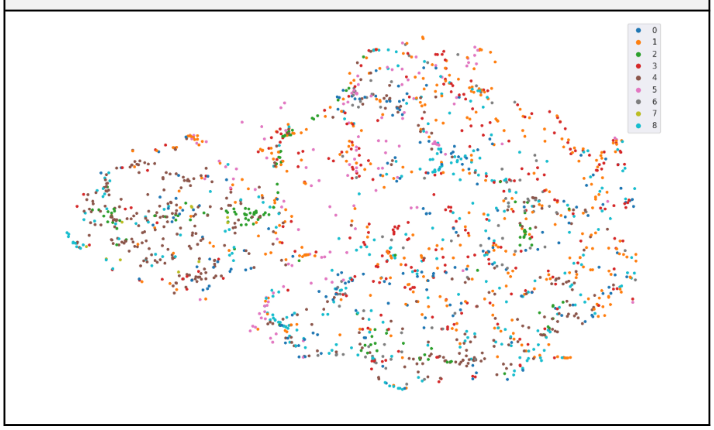


Sensor Data Analysis

The heat map of the normalized confusion matrix

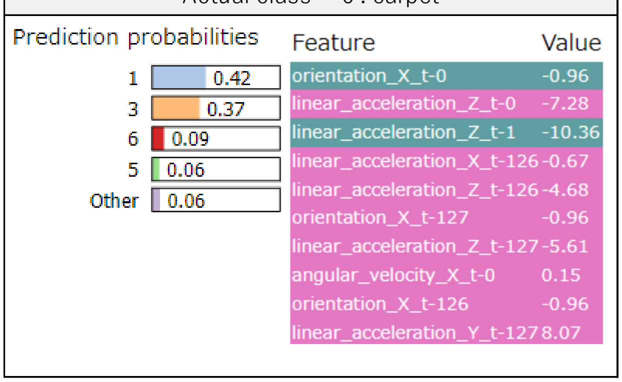


t-SNE plot



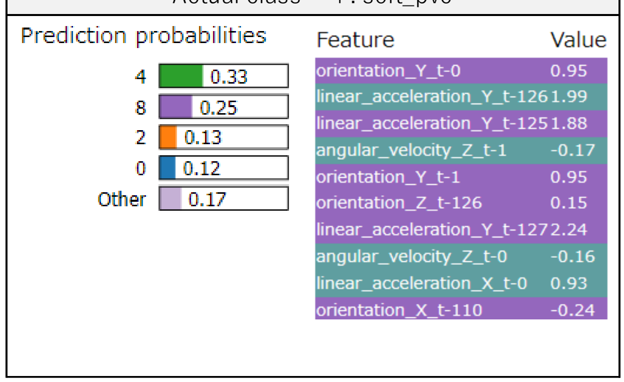
Lime explanation (prediction for series_id = 1)

Actual class = 6 : carpet

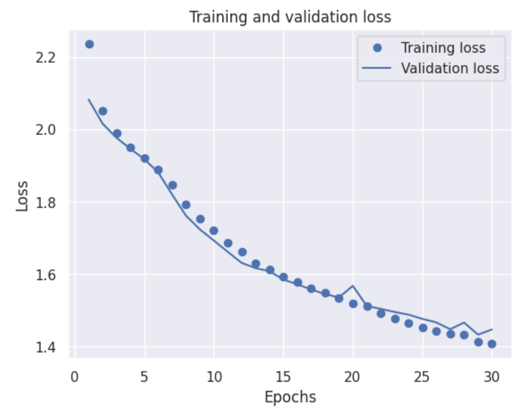
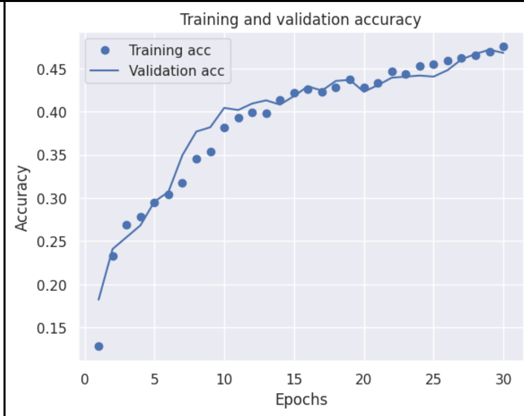


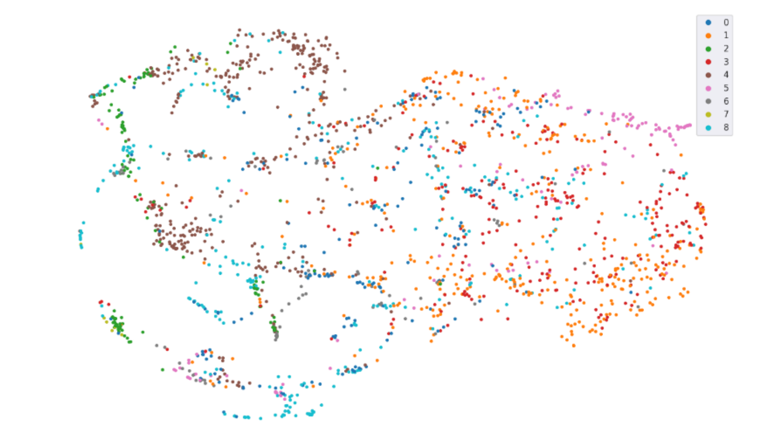
Lime explanation (prediction for series_id = 4)

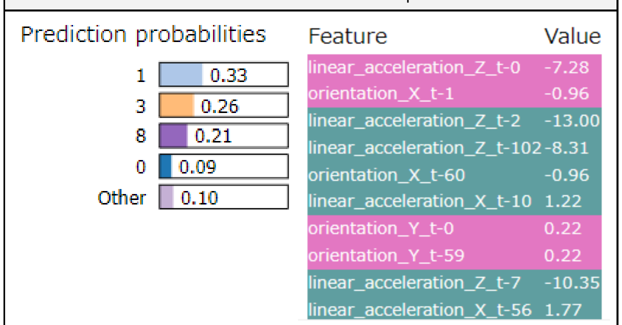
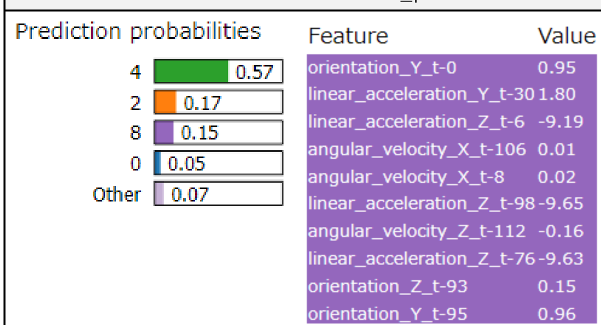
Actual class = 4 : soft_pvc



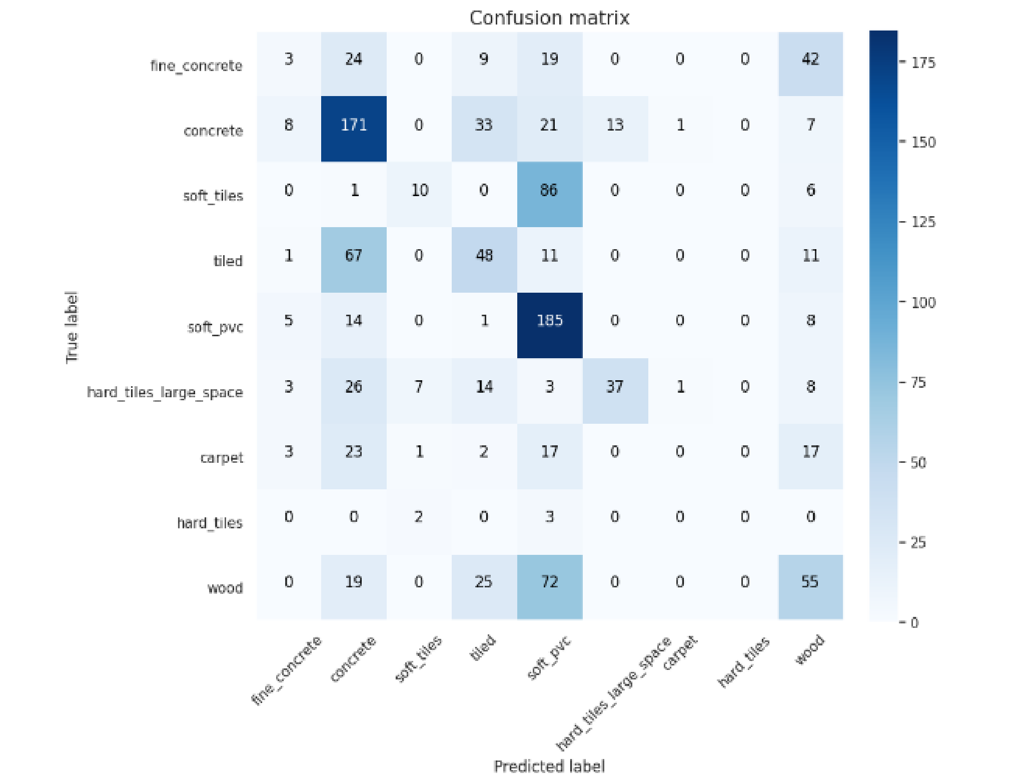
Experiment B-3

ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
B-3		

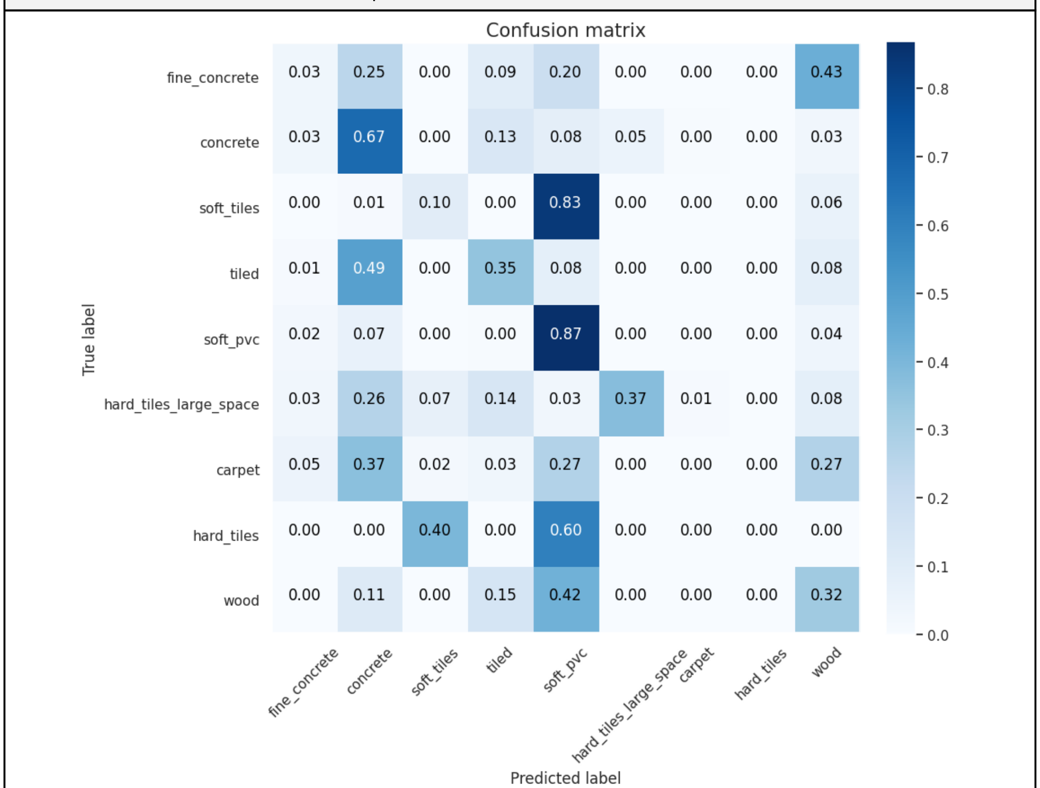
t-SNE plot


Lime explanation (prediction for series_id = 1) Actual class = 6 : carpet	Lime explanation (prediction for series_id = 4) Actual class = 4 : soft_pvc
	

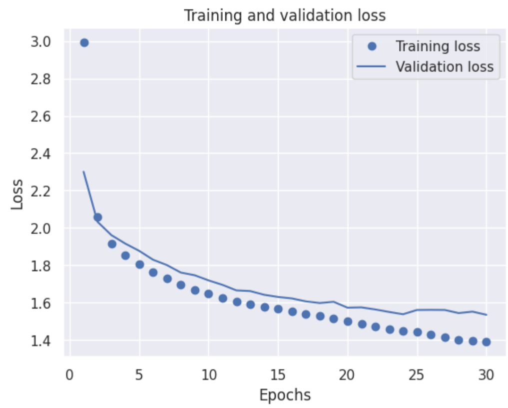
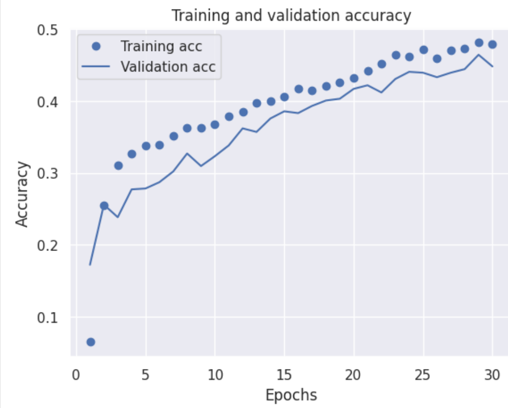
The heat map of the confusion matrix

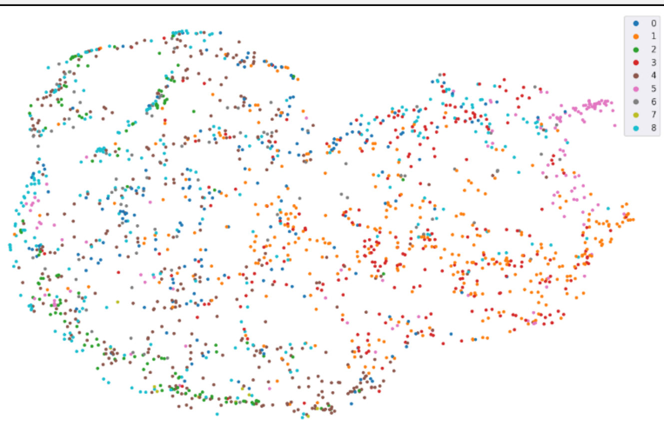


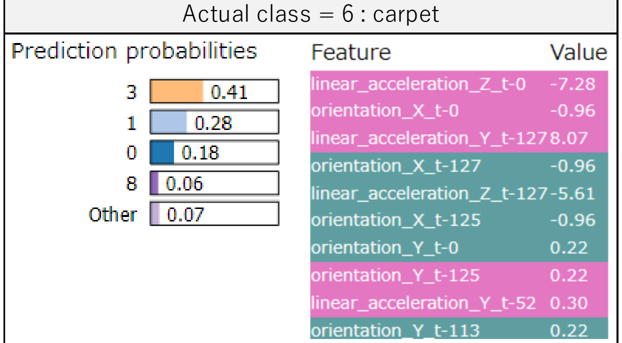
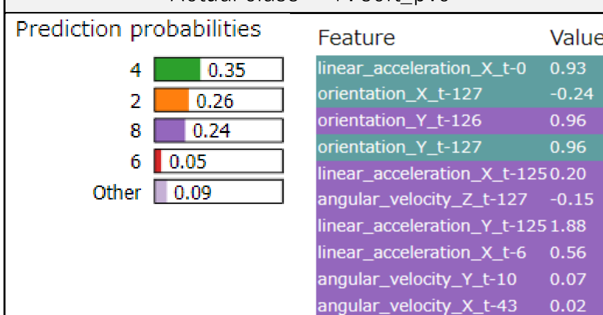
The heat map of the normalized confusion matrix



Experiment B-4

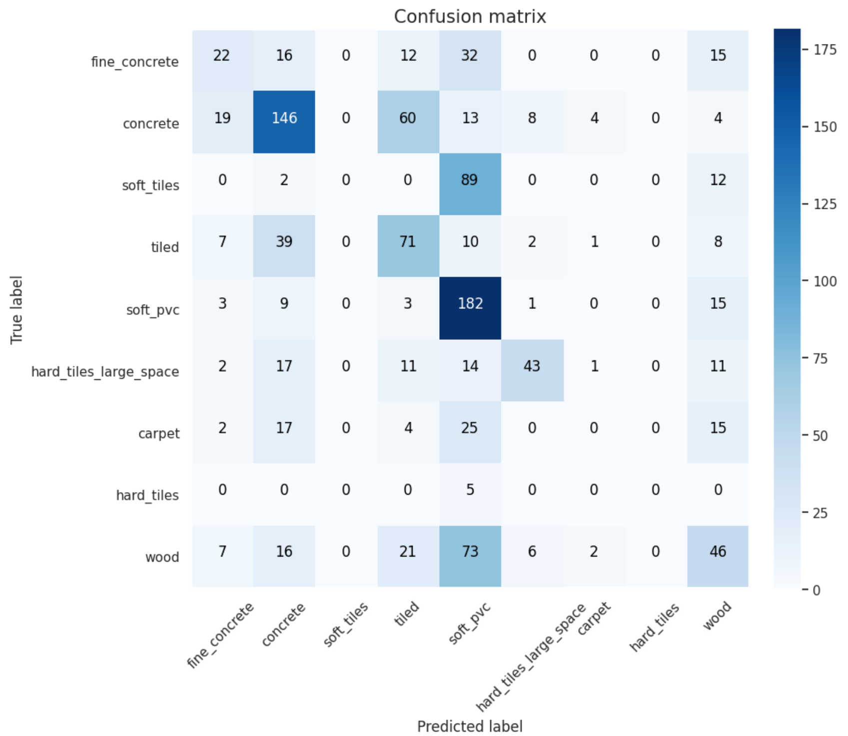
ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
B-4		

t-SNE plot


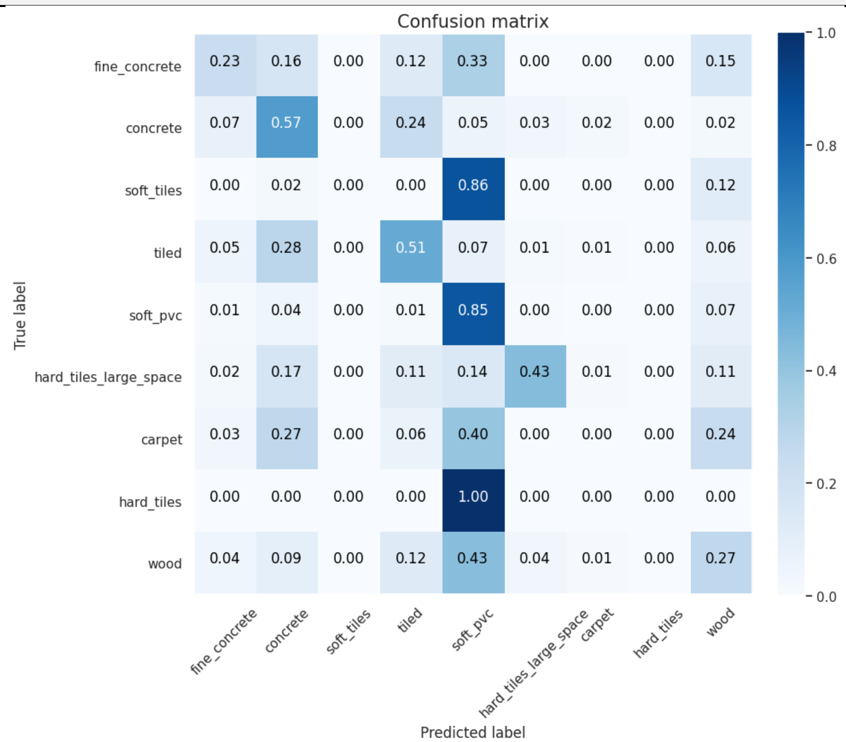
Lime explanation (prediction for series_id = 1) Actual class = 6 : carpet	Lime explanation (prediction for series_id = 4) Actual class = 4 : soft_pvc
	

Sensor Data Analysis

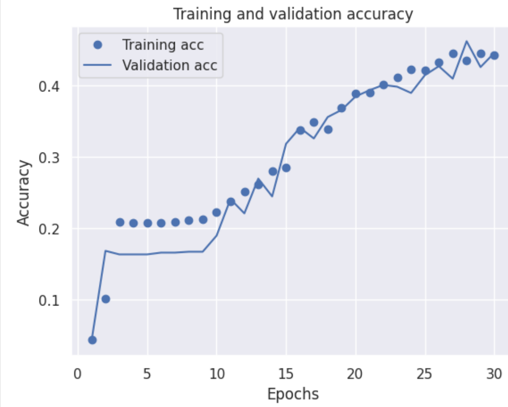
The heat map of the confusion matrix

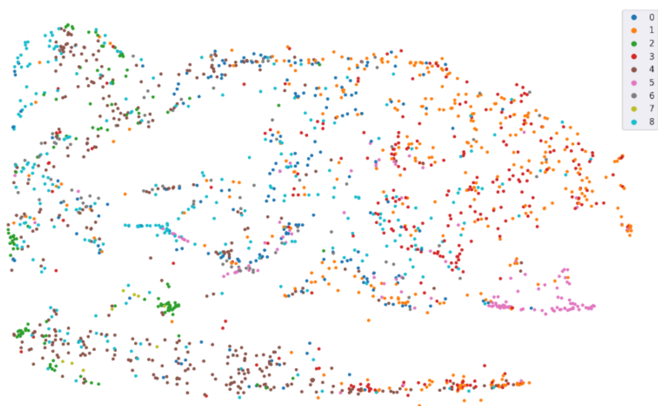


The heat map of the normalized confusion matrix



Experiment B-5

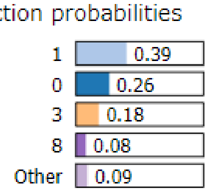
ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
B-5		

t-SNE plot


Lime explanation (prediction for series_id = 1)

Actual class = 6 : carpet

Prediction probabilities

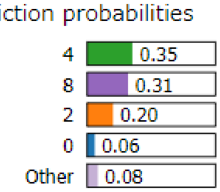


Feature	Value
orientation_X_t-1	-0.96
linear_acceleration_Z_t-0	-7.28
orientation_X_t-0	-0.96
orientation_X_t-2	-0.96
linear_acceleration_X_t-1	0.83
orientation_Y_t-3	0.22
orientation_X_t-3	-0.96
orientation_Y_t-1	0.22
linear_acceleration_X_t-47	-1.43
orientation_Y_t-107	0.22

Lime explanation (prediction for series_id = 4)

Actual class = 4 : soft_pvc

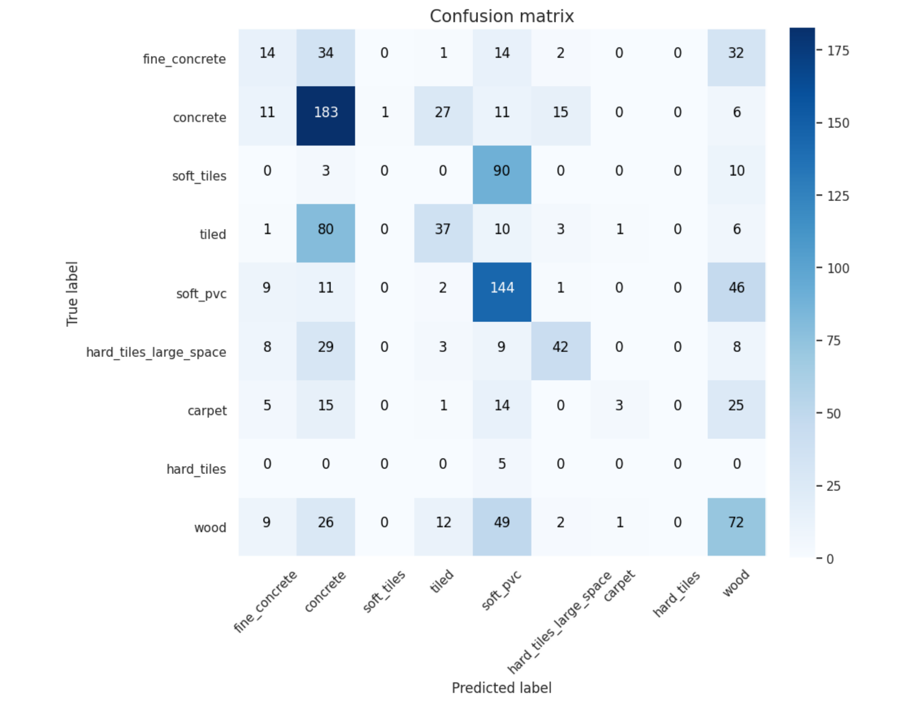
Prediction probabilities



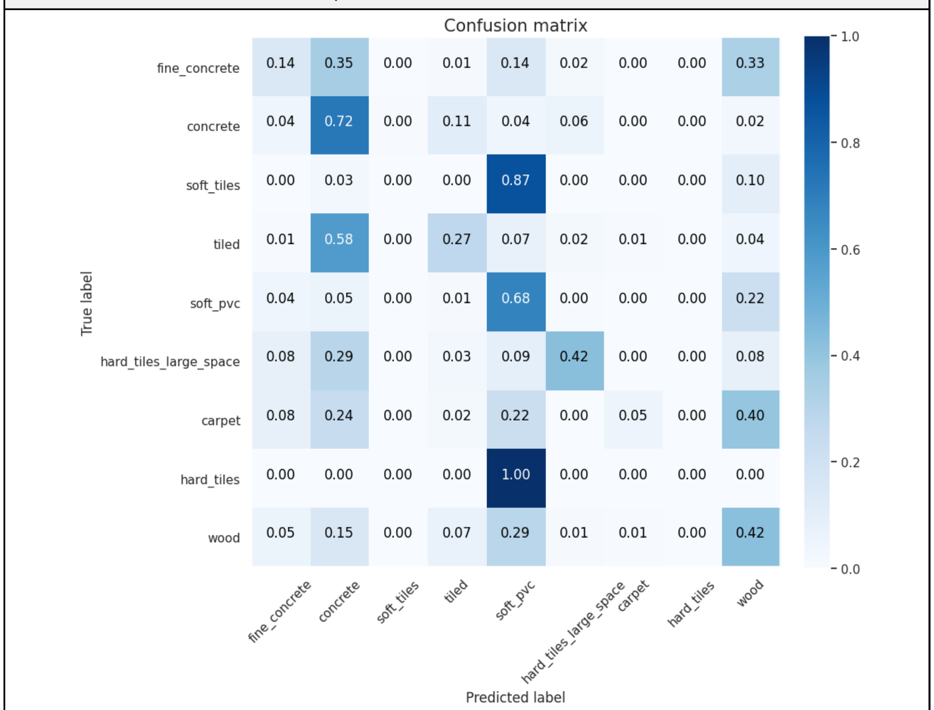
Feature	Value
linear_acceleration_X_t-0	0.93
orientation_Y_t-1	0.95
orientation_X_t-0	-0.26
orientation_X_t-1	-0.26
linear_acceleration_X_t-16	0.37
orientation_W_t-47	-0.05
orientation_Y_t-3	0.95
linear_acceleration_X_t-106	0.15
orientation_X_t-114	-0.24
orientation_Y_t-5	0.95

Sensor Data Analysis

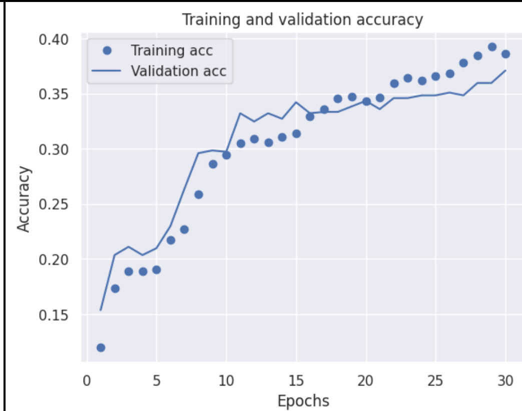
The heat map of the confusion matrix

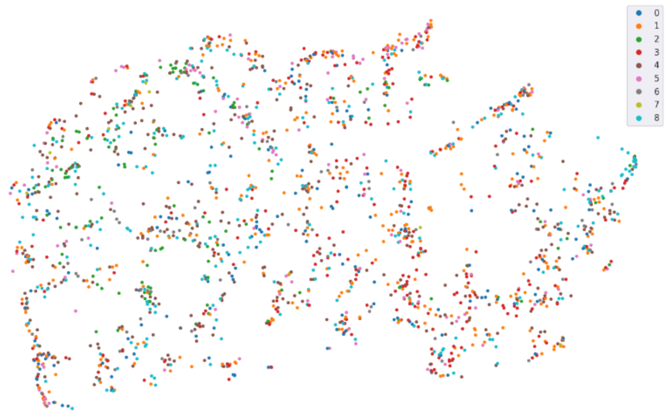


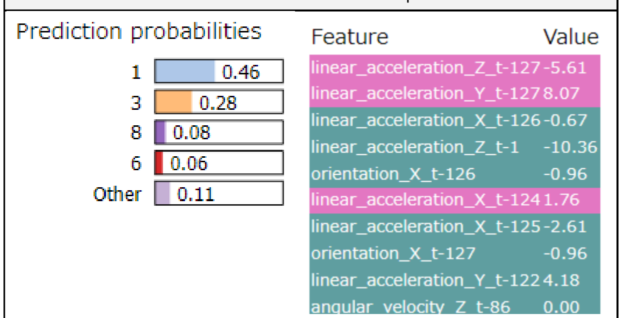
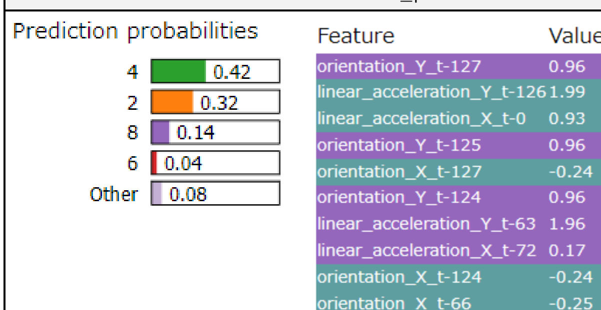
The heat map of the normalized confusion matrix



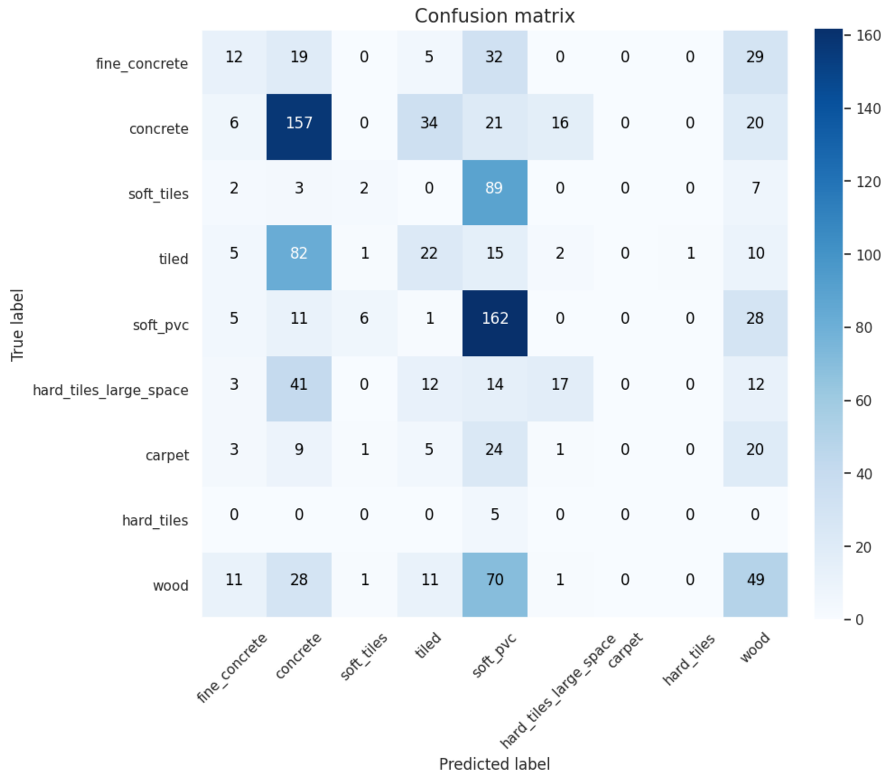
Experiment B-6

ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
B-6		

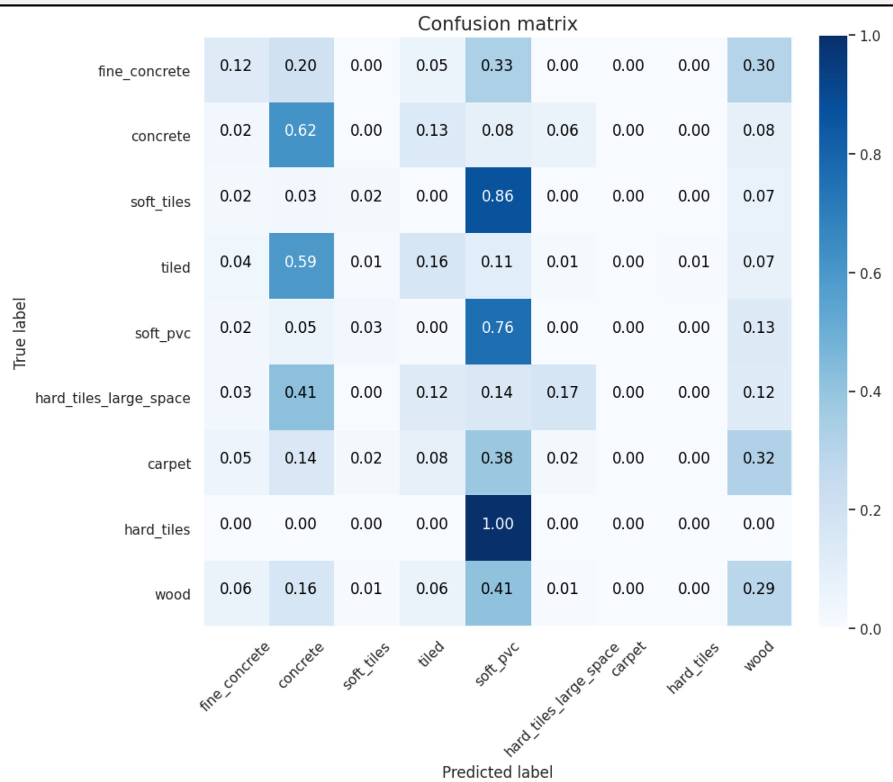
t-SNE plot


Lime explanation (prediction for series_id = 1) Actual class = 6 : carpet	Lime explanation (prediction for series_id = 4) Actual class = 4 : soft_pvc
	

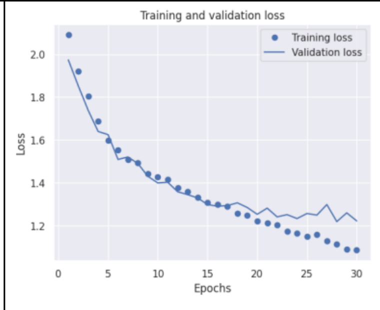
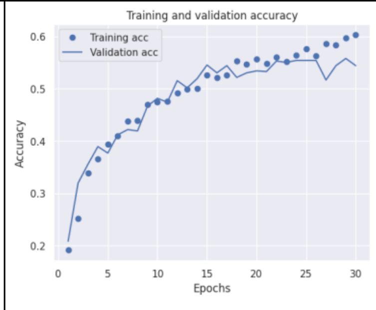
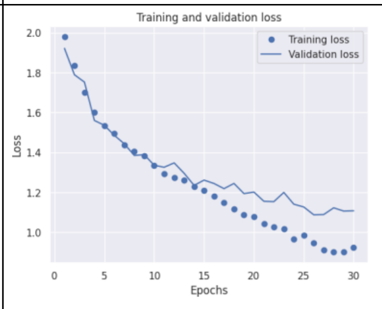
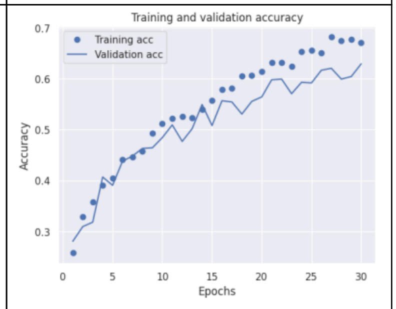
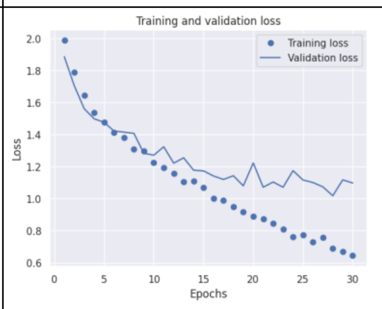
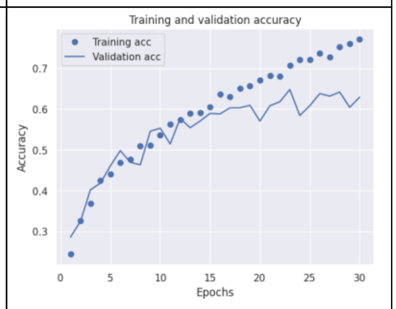
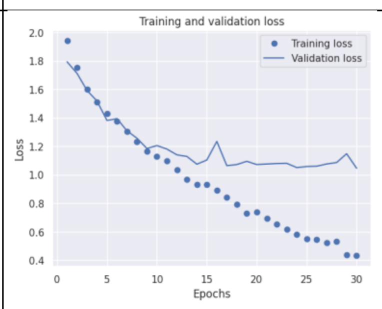
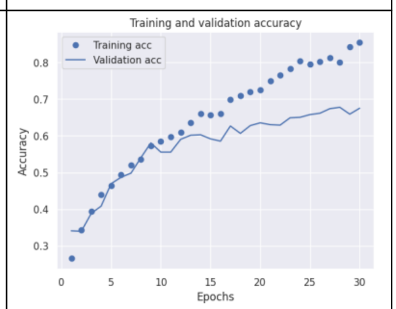
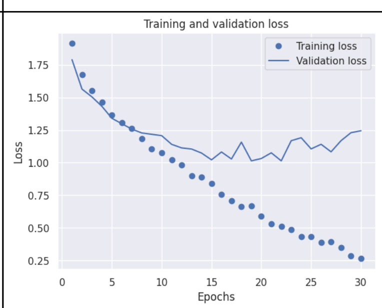
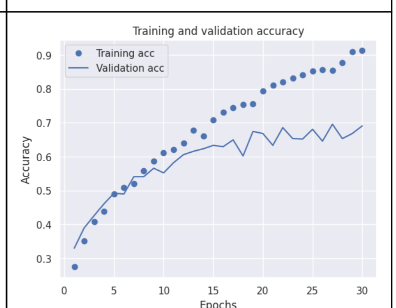
The heat map of the confusion matrix



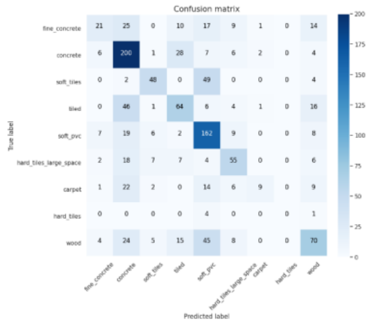
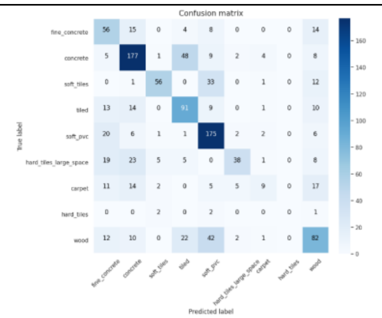
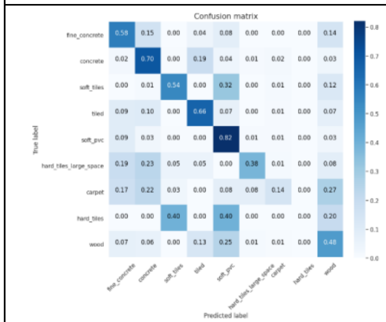
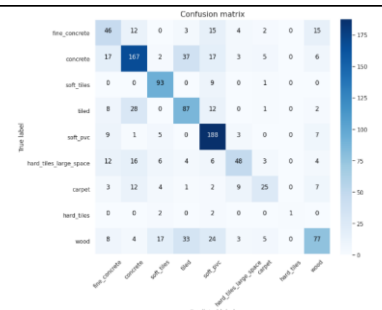
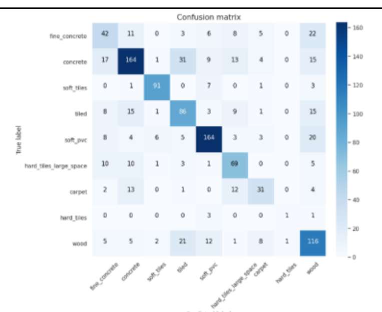
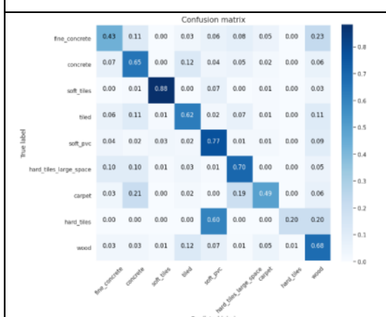
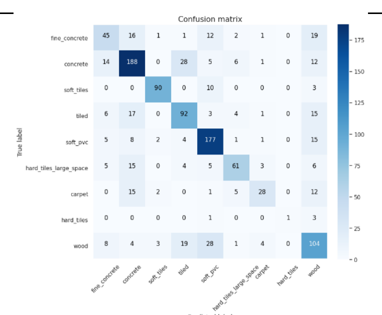
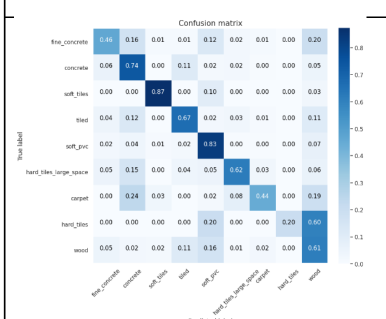
The heat map of the normalized confusion matrix



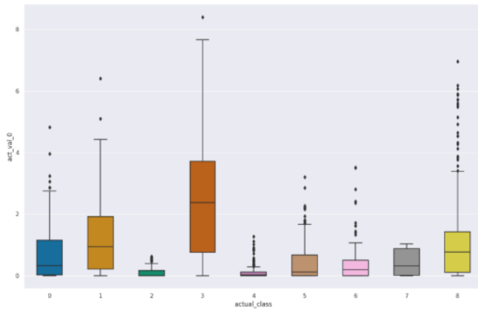
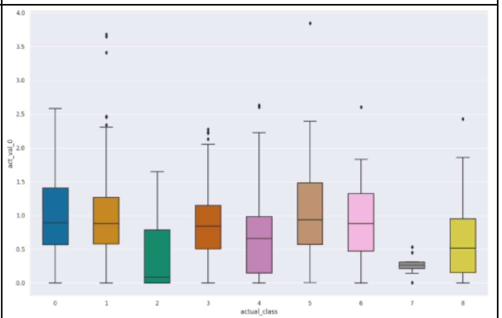
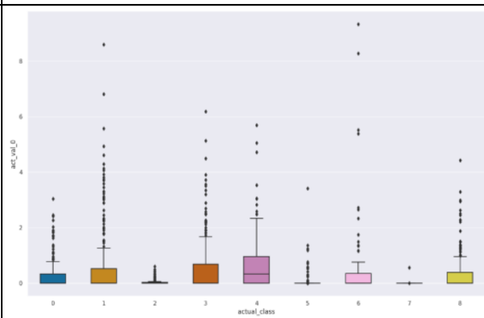
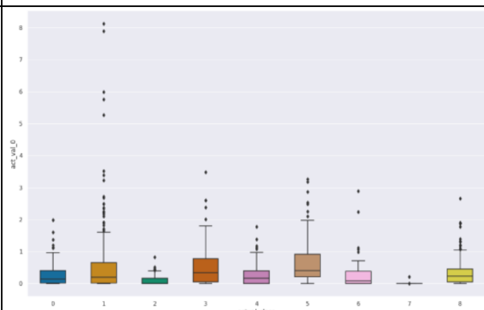
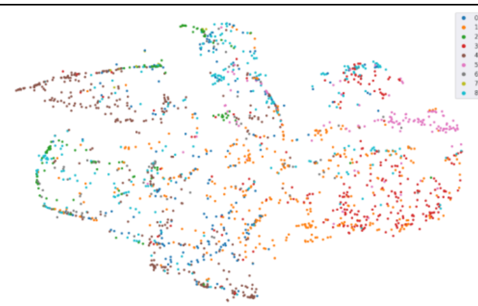
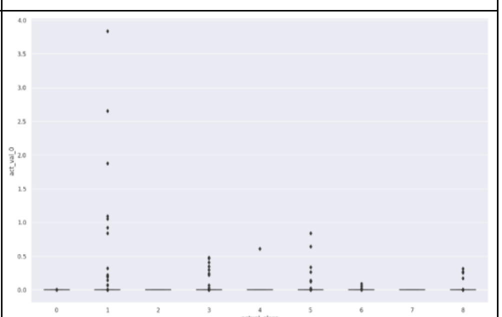
Experiment C

ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
C-1		
C-2		
C-3		
C-4		
C-5		

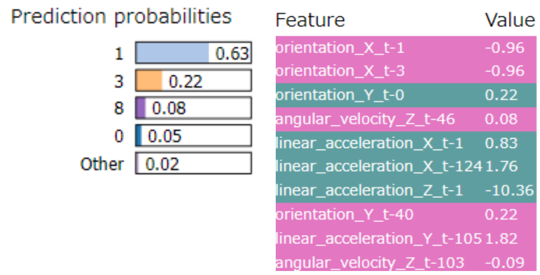
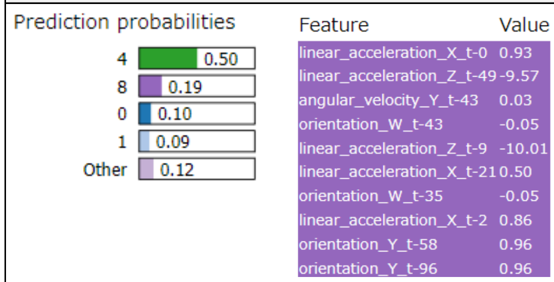
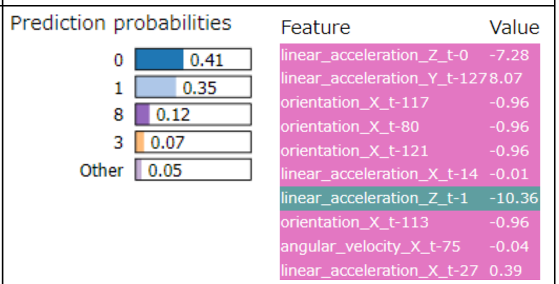
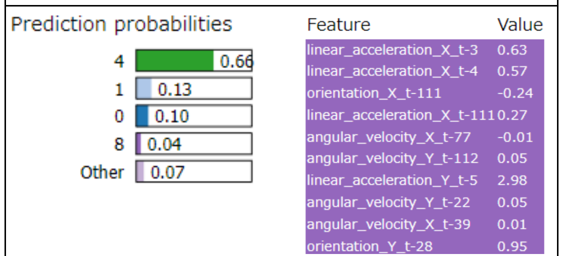
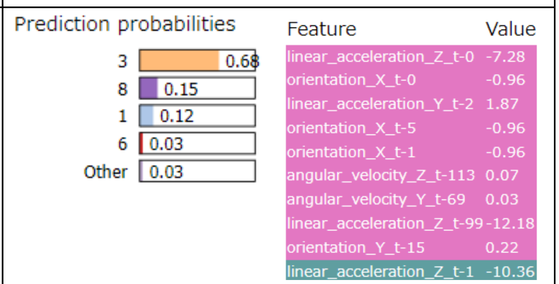
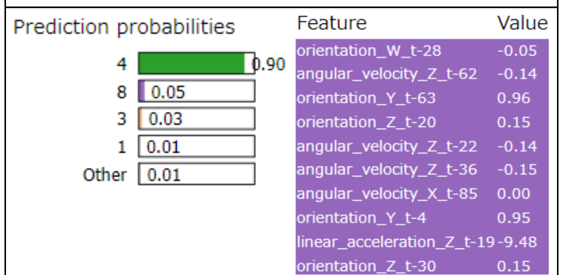
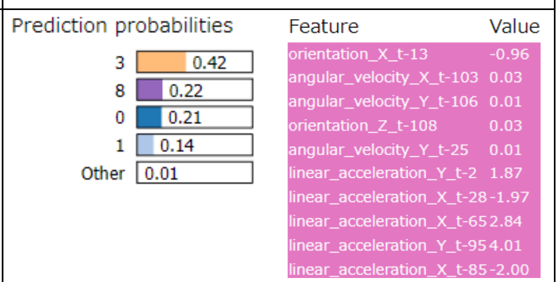
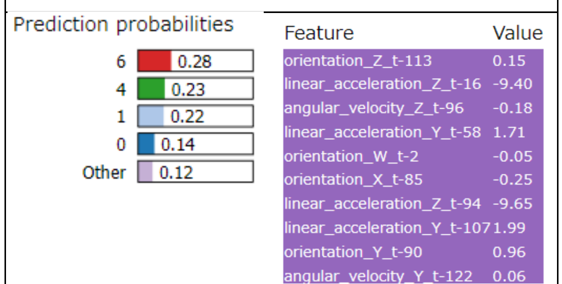
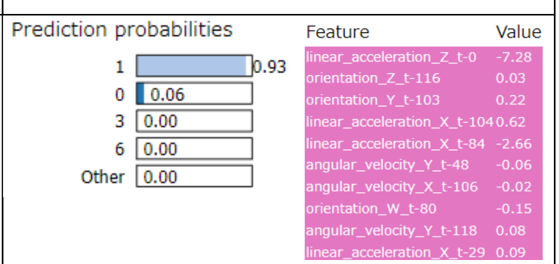
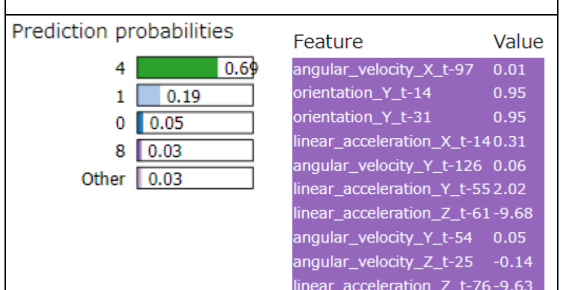
Sensor Data Analysis

ModelNum	The heat map of the confusion matrix	The heat map of the normalized confusion matrix
C-1		
C-2		
C-3		
C-4		
C-5		

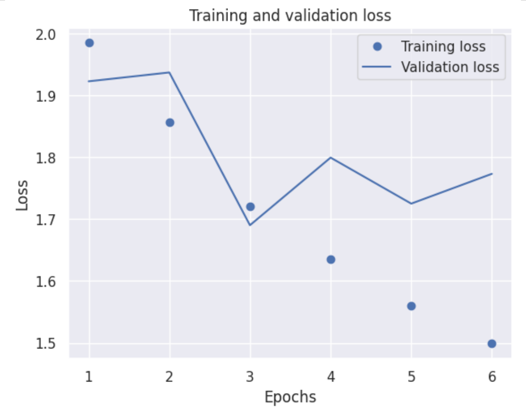
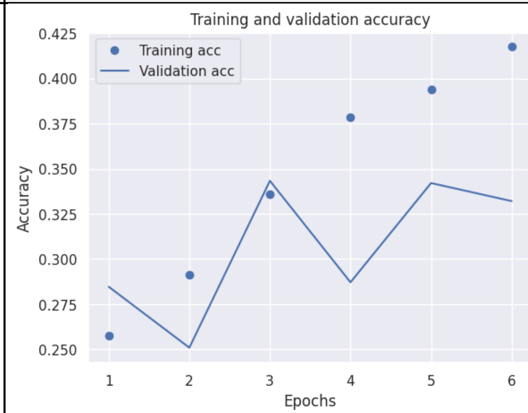
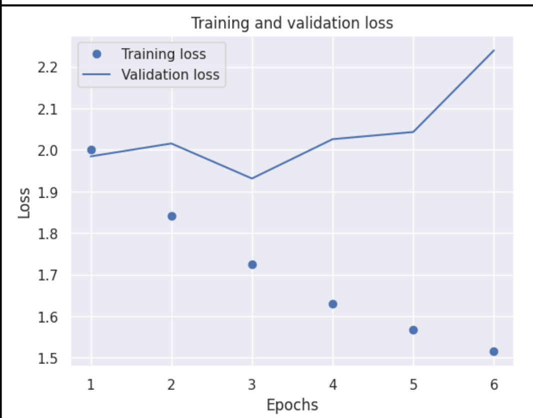
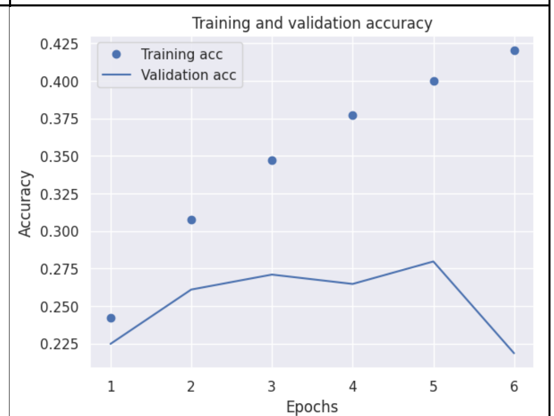
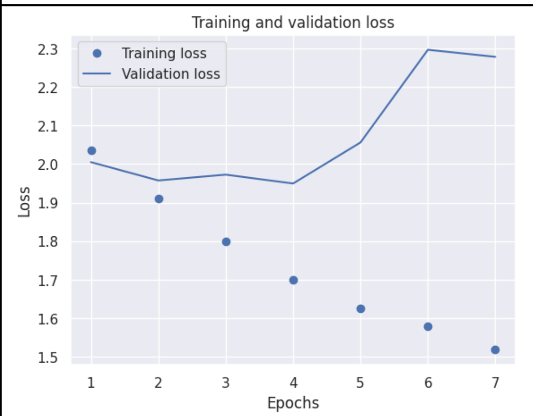
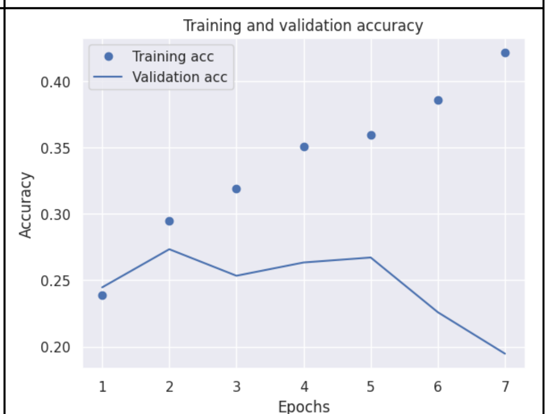
Sensor Data Analysis

ModelNum	Activation value[0] for each class	t-SNE plot
C-1		
C-2		
C-3		
C-4		
C-5		

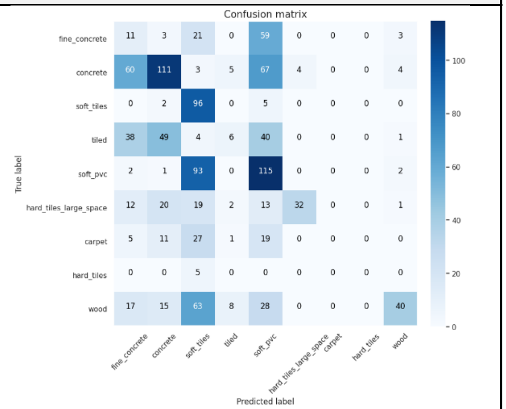
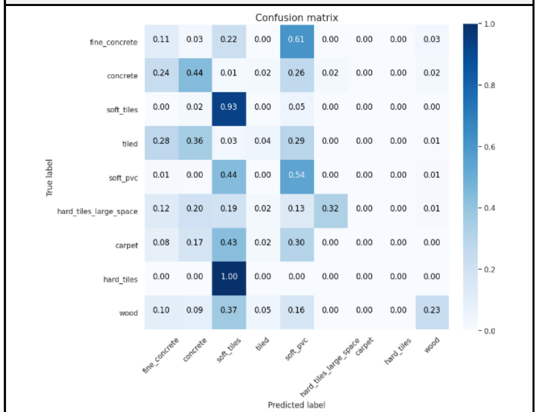
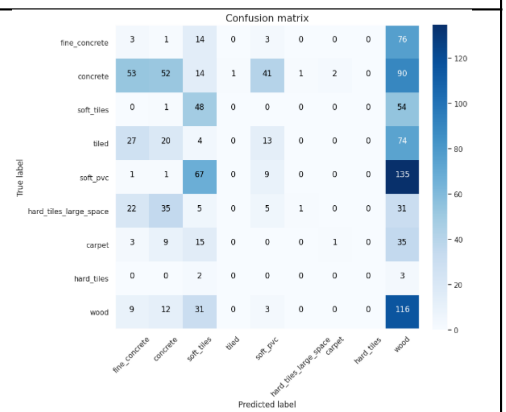
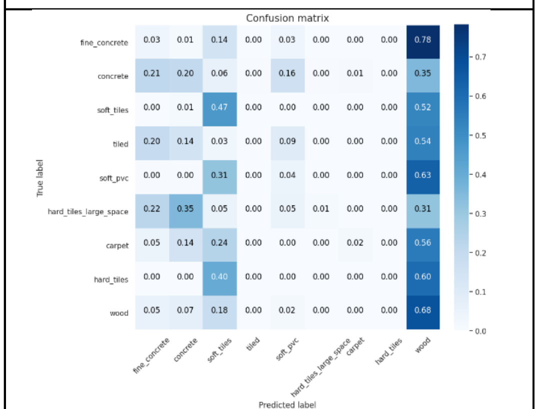
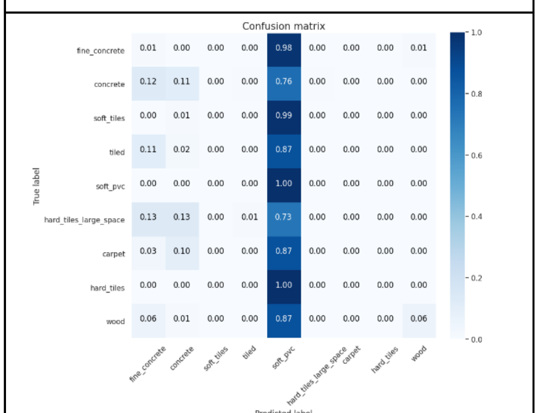
Sensor Data Analysis

<table>
<tr><th>ModelNum</th><th>Lime explanation (prediction for series_id = 1)
Actual class = 6 : carpet</th><th>Lime explanation (prediction for series_id = 4)
Actual class = 4 : soft_pvc</th></tr>
<tr><td>C-1</td><td></td><td></td></tr>
<tr><td>C-2</td><td></td><td></td></tr>
<tr><td>C-3</td><td></td><td></td></tr>
<tr><td>C-4</td><td></td><td></td></tr>
<tr><td>C-5</td><td></td><td></td></tr>
</table>

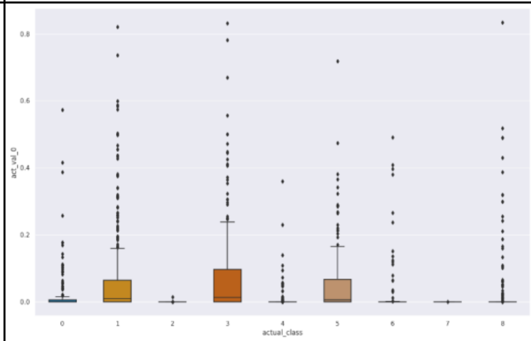
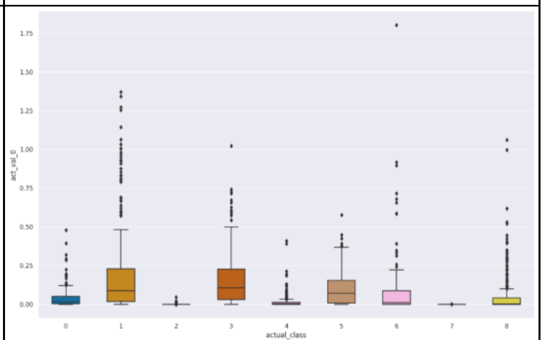
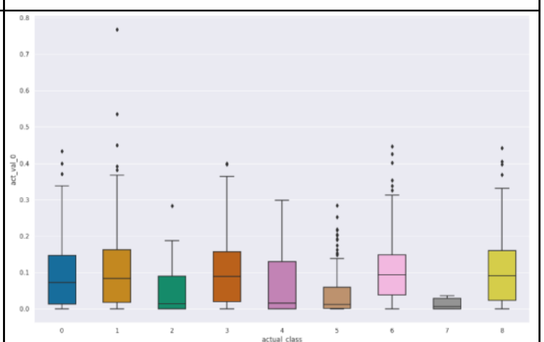
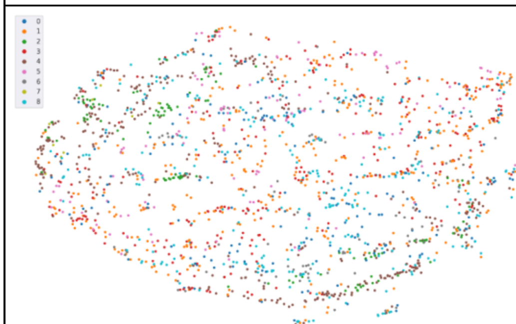
Experiment E

<table>
<tr><th>ModelNum</th><th>The training and validation loss for each training epoch</th><th>The training and validation accuracy for each training epoch</th></tr>
<tr><td>E-1</td><td></td><td></td></tr>
<tr><td>E-2</td><td></td><td></td></tr>
<tr><td>E-3</td><td></td><td></td></tr>
</table>

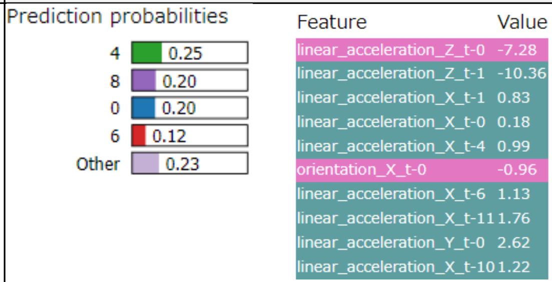
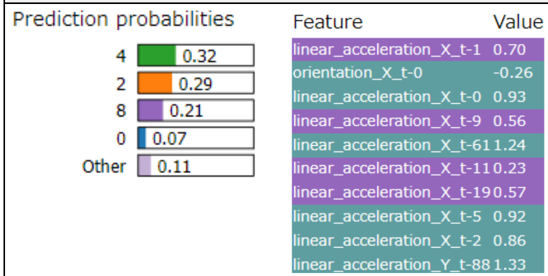
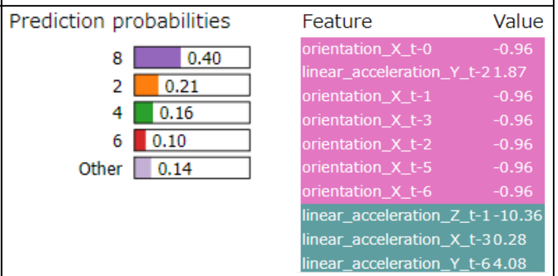
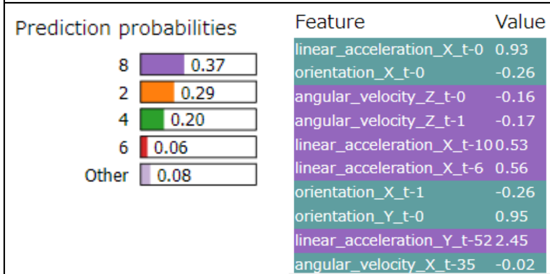
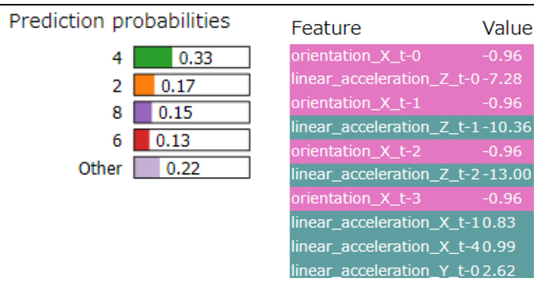
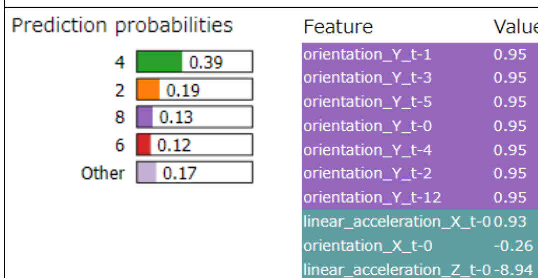
Sensor Data Analysis

ModelNum	The heat map of the confusion matrix	The heat map of the normalized confusion matrix
E-1		
E-2		
E-3		

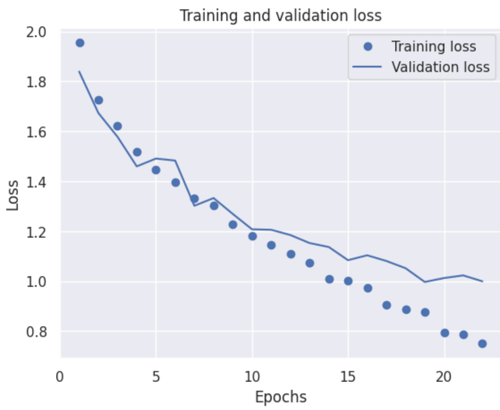
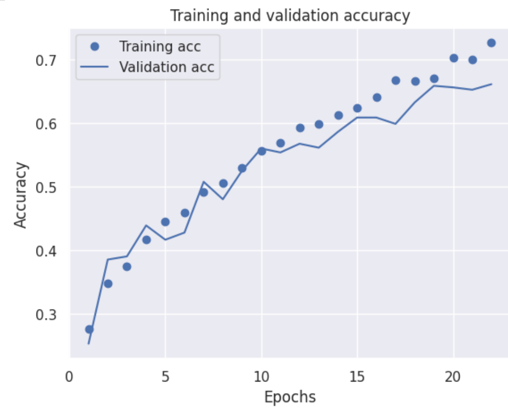
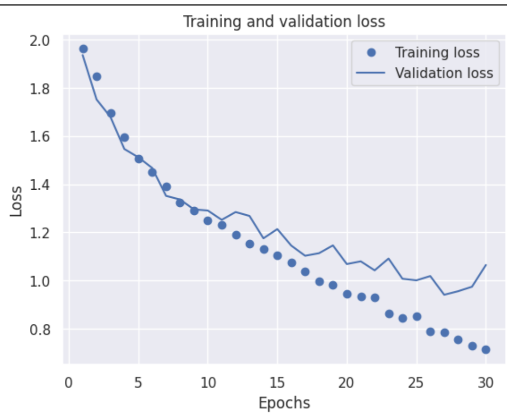
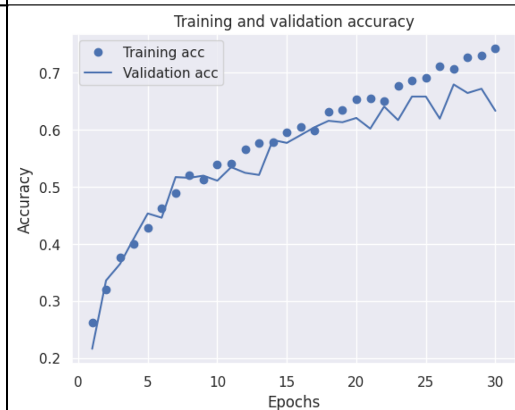
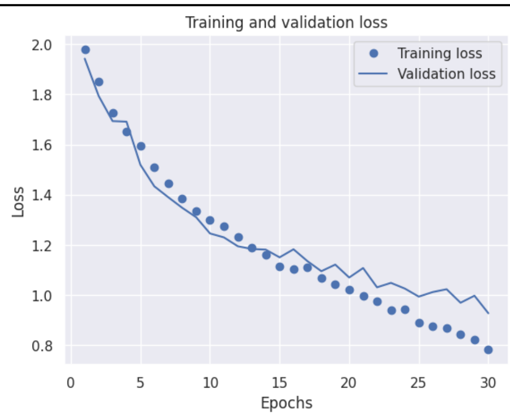
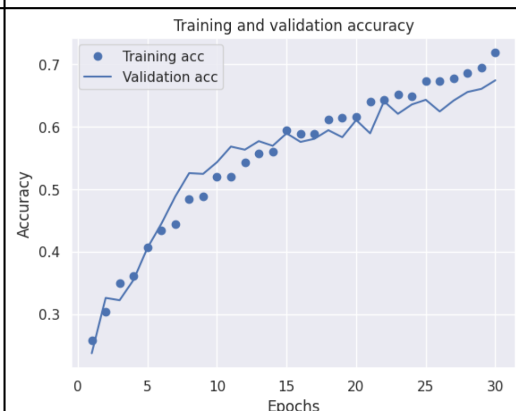
Sensor Data Analysis

ModelNum	Activation value[0] for each class	t-SNE plot
E-1		
E-2		
E-3		

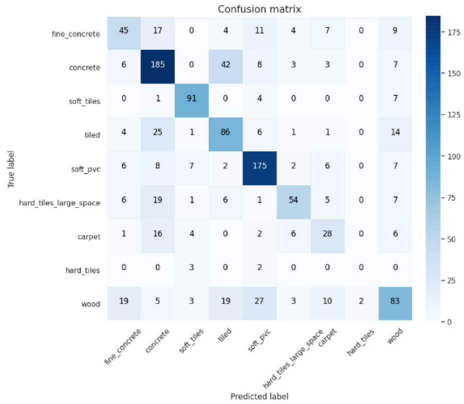
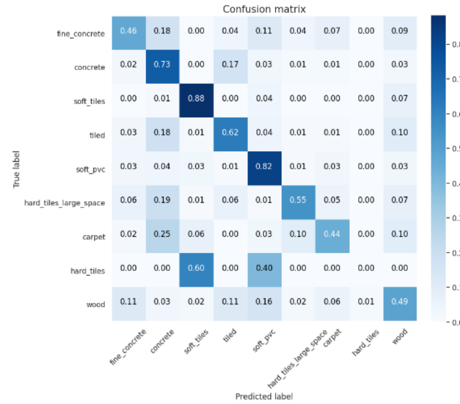
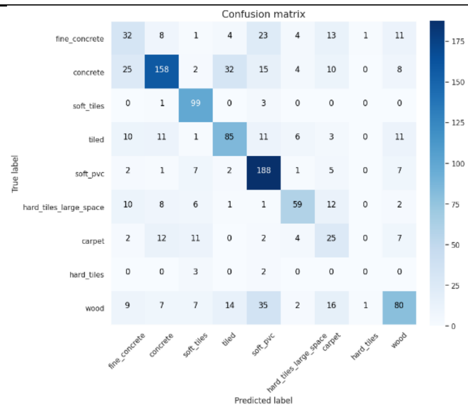
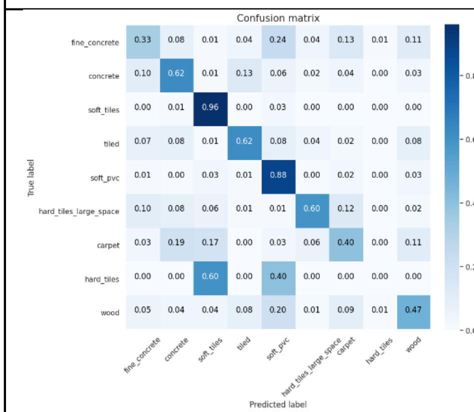
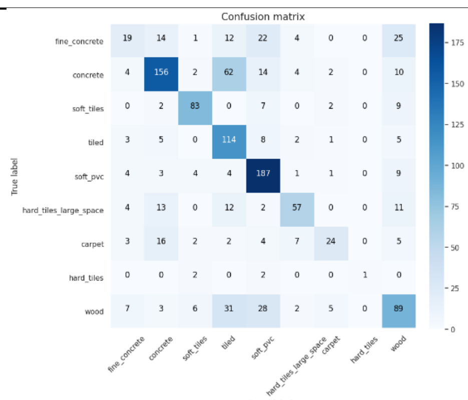
Sensor Data Analysis

<table>
<tr><th>ModelNum</th><th>Lime explanation (prediction for series_id = 1)
Actual class = 6 : carpet</th><th>Lime explanation (prediction for series_id = 4)
Actual class = 4 : soft_pvc</th></tr>
<tr><td>E-1</td><td>
</td><td>
</td></tr>
<tr><td>E-2</td><td>
</td><td>
</td></tr>
<tr><td>E-3</td><td>
</td><td>
</td></tr>
</table>

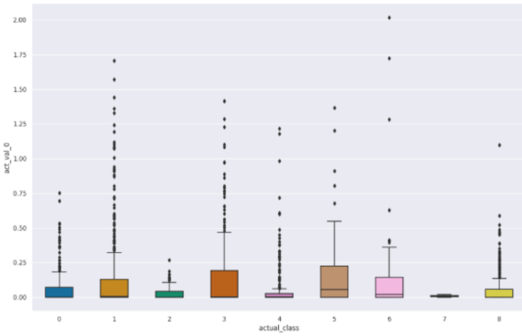
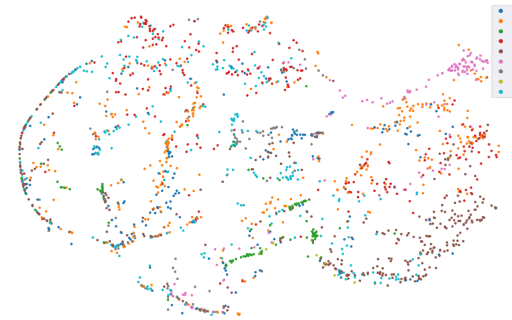
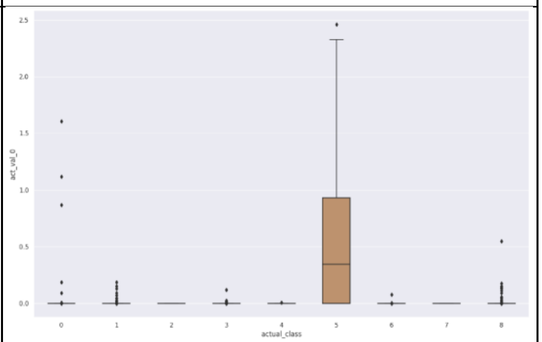
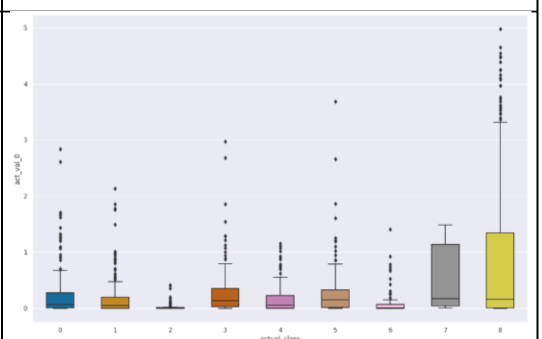
Experiment F

ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
F-1		
F-2		
F-3		

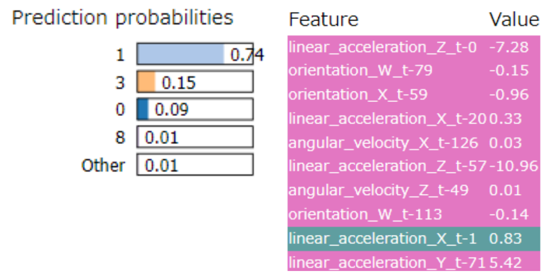
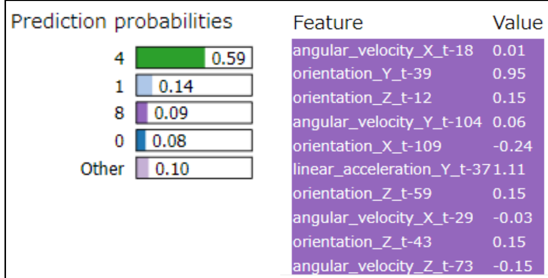
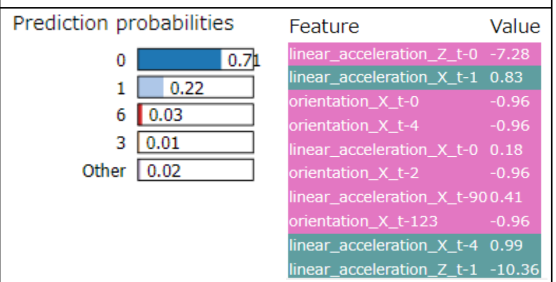
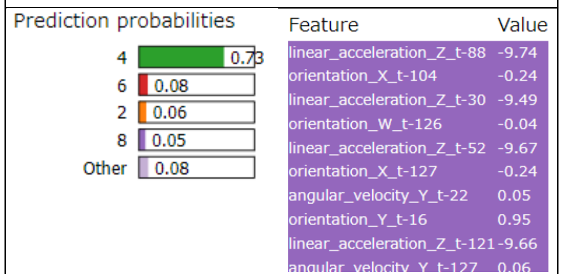
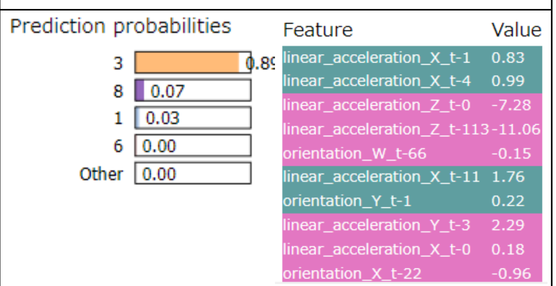
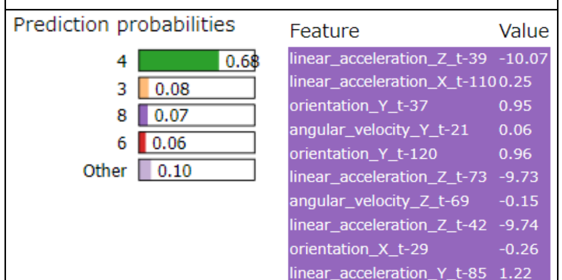
Sensor Data Analysis

<table>
<tr><th>ModelNum</th><th>The heat map of the confusion matrix</th><th>The heat map of the normalized confusion matrix</th></tr>
<tr><td>F-1</td><td>
</td><td>
</td></tr>
<tr><td>F-2</td><td>
</td><td>
</td></tr>
<tr><td>F-3</td><td>
</td><td>
</td></tr>
</table>

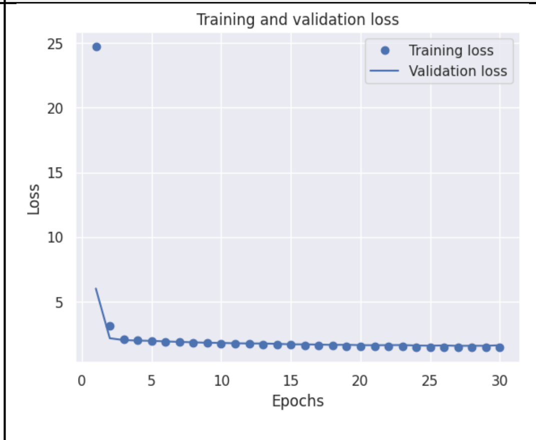
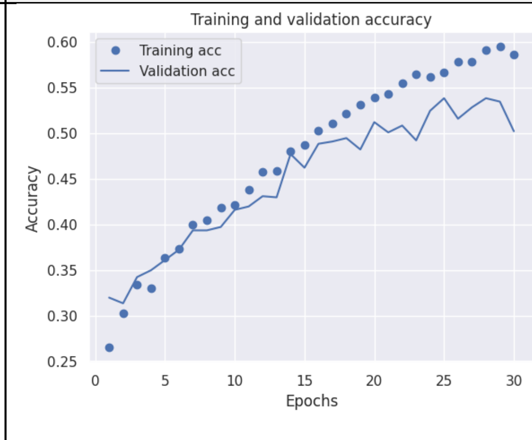
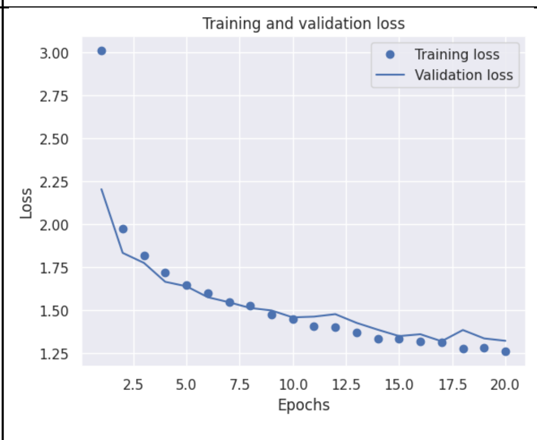
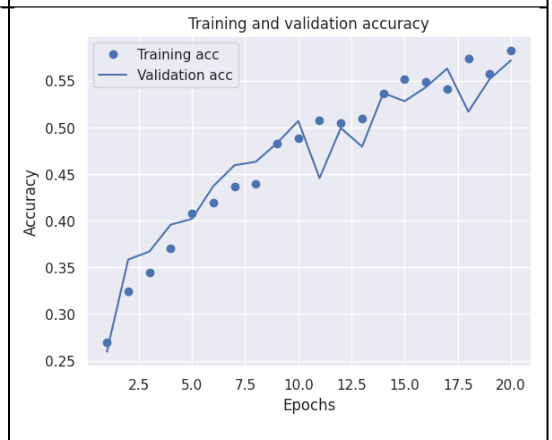
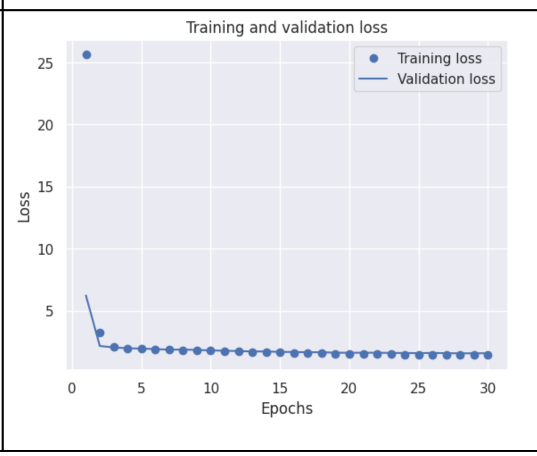
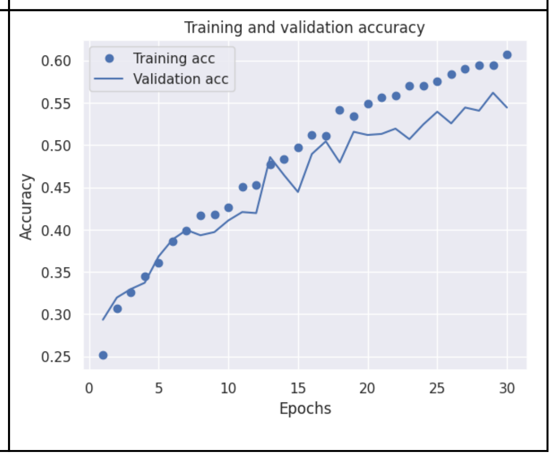
Sensor Data Analysis

ModelNum	Activation value[0] for each class	t-SNE plot
F-1		
F-2		
F-3		

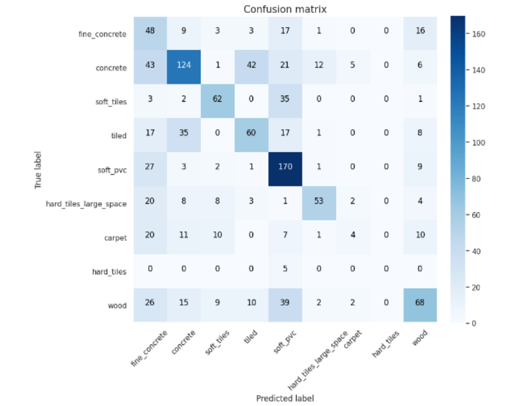
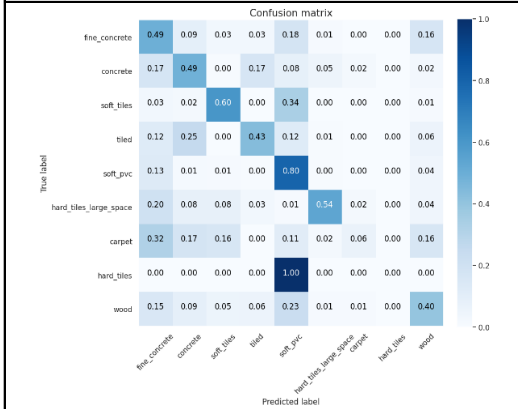
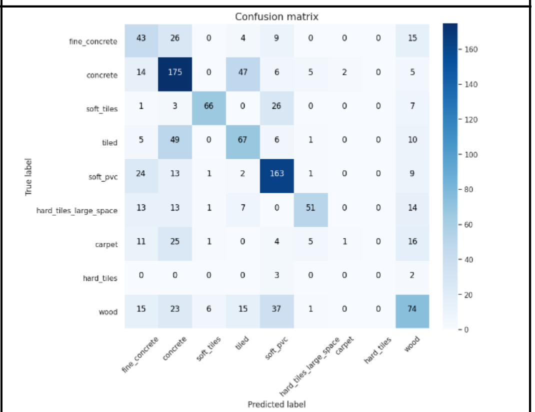
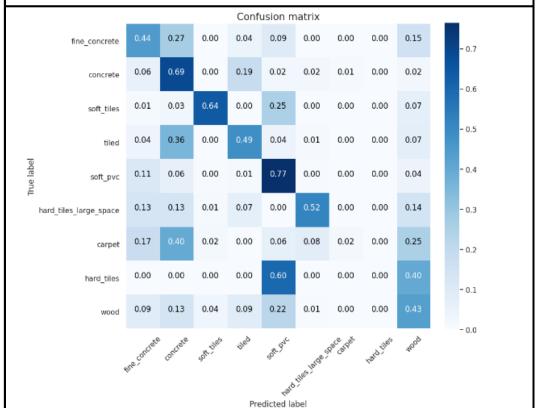
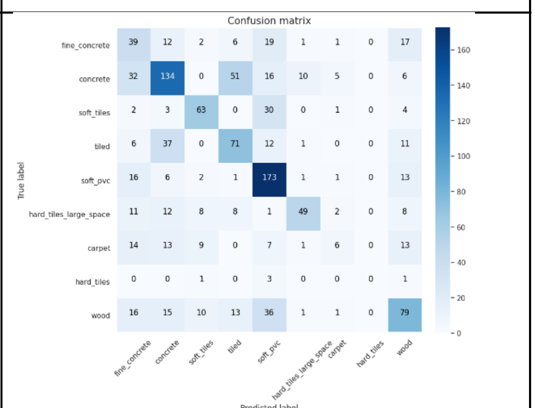
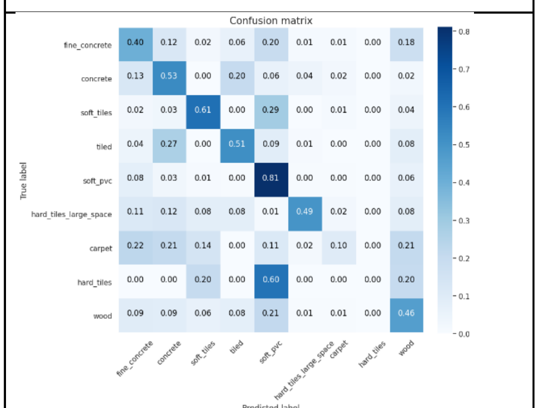
Sensor Data Analysis

<table>
<tr><th>ModelNum</th><th>Lime explanation (prediction for series_id = 1)
Actual class = 6 : carpet</th><th>Lime explanation (prediction for series_id = 4)
Actual class = 4 : soft_pvc</th></tr>
<tr><td>F-1</td><td>
</td><td>
</td></tr>
<tr><td>F-2</td><td>
</td><td>
</td></tr>
<tr><td>F-3</td><td>
</td><td>
</td></tr>
</table>

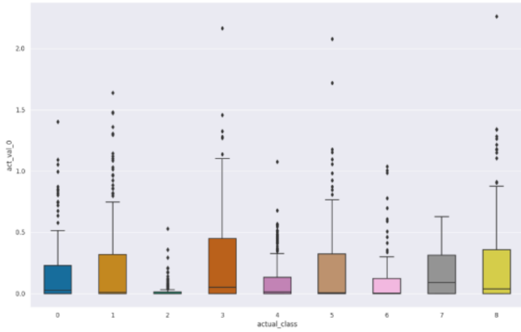
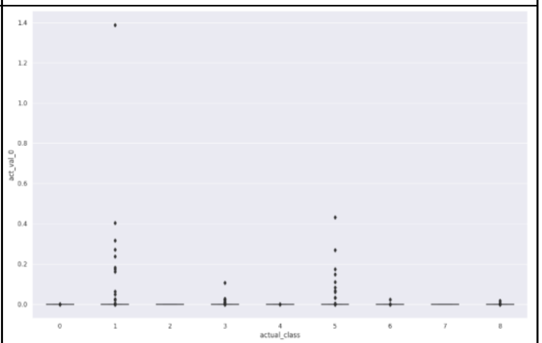
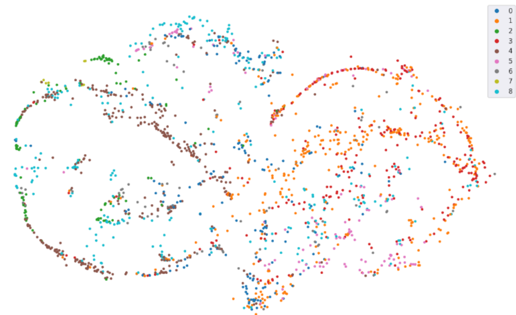
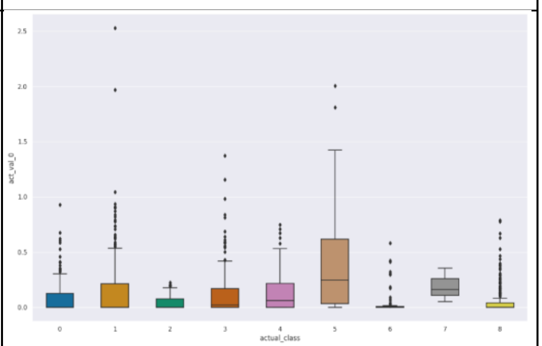
Experiment G

ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
G-1		
G-2		
G-3		

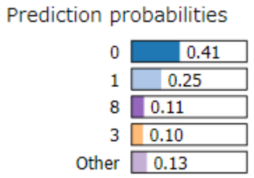
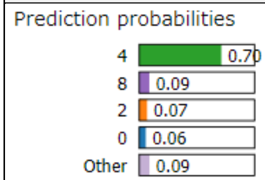
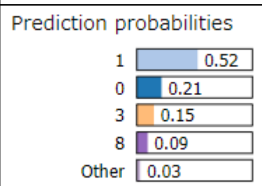
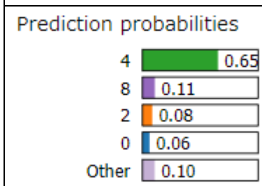
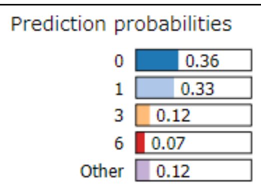
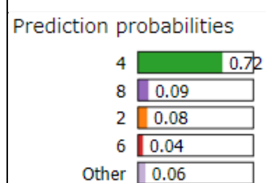
Sensor Data Analysis

ModelNum	The heat map of the confusion matrix	The heat map of the normalized confusion matrix
G-1		
G-2		
G-3		

Sensor Data Analysis

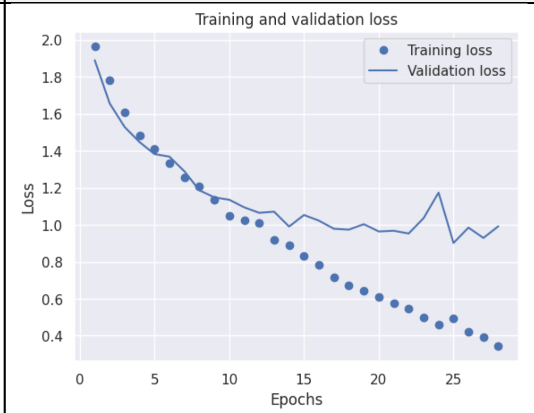
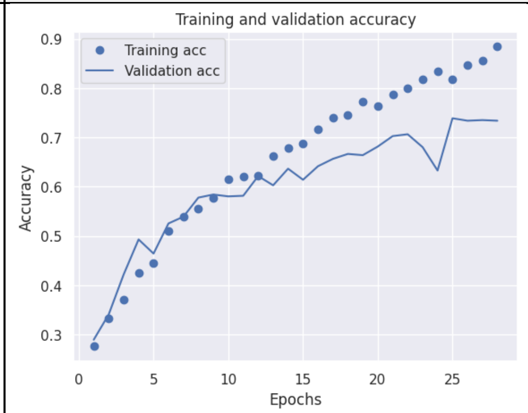
ModelNum	Activation value[0] for each class	t-SNE plot
G-1		
G-2		
G-3		

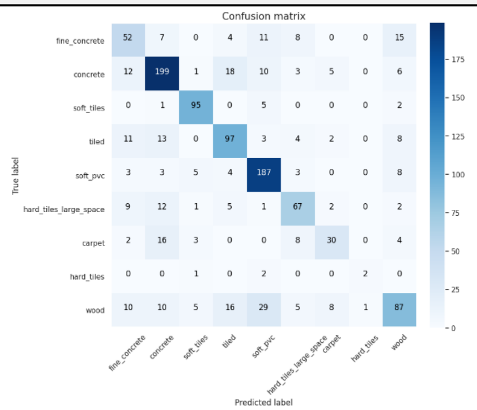
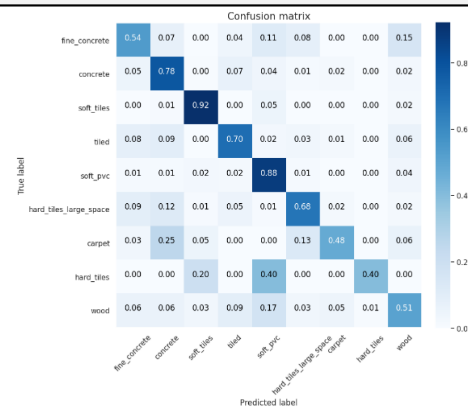
Sensor Data Analysis

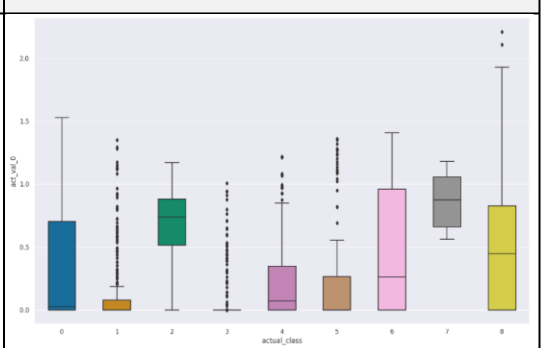
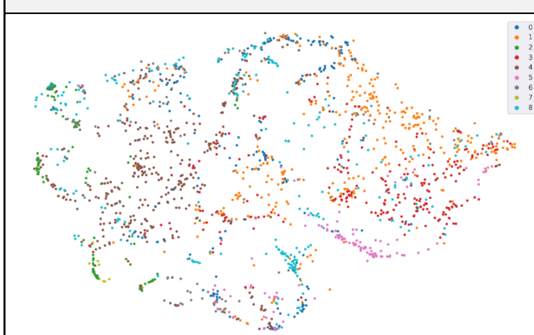
ModelNum	Lime explanation (prediction for series_id = 1) Actual class = 6 : carpet	Lime explanation (prediction for series_id = 4) Actual class = 4 : soft_pvc
G-1	 Feature Value orientation_X_t-0 -0.96 linear_acceleration_Z_t-0 -7.28 angular_velocity_X_t-0 0.15 linear_acceleration_Y_t-6 4.08 angular_velocity_X_t-59 0.06 linear_acceleration_Z_t-1 -10.36 linear_acceleration_X_t-0 0.18 orientation_X_t-1 -0.96 orientation_Y_t-96 0.22 orientation_W_t-48 -0.14	 Feature Value linear_acceleration_X_t-0 0.93 orientation_X_t-0 -0.26 linear_acceleration_X_t-1 0.70 linear_acceleration_Z_t-3 -10.11 orientation_Z_t-0 0.15 angular_velocity_X_t-12 -0.01 linear_acceleration_Z_t-100 -9.77 angular_velocity_X_t-52 0.01 linear_acceleration_X_t-91 0.39 linear_acceleration_Z_t-97 -9.78
G-2	 Feature Value linear_acceleration_Z_t-0 -7.28 linear_acceleration_Z_t-1 -10.36 orientation_Y_t-0 0.22 linear_acceleration_Z_t-2 -13.00 orientation_W_t-113 -0.14 linear_acceleration_Y_t-0 2.62 orientation_W_t-8 -0.14 linear_acceleration_X_t-1 0.83 angular_velocity_Z_t-42 0.10 orientation_Z_t-4 0.03	 Feature Value linear_acceleration_Y_t-15 2.16 linear_acceleration_Z_t-14 -9.51 angular_velocity_Z_t-54 -0.14 linear_acceleration_X_t-15 0.20 orientation_X_t-108 -0.24 linear_acceleration_Z_t-21 -9.67 angular_velocity_Z_t-51 -0.14 orientation_Y_t-3 0.95 linear_acceleration_X_t-74 0.19 orientation_X_t-0 -0.26
G-3	 Feature Value orientation_X_t-0 -0.96 linear_acceleration_Z_t-0 -7.28 linear_acceleration_X_t-0 0.18 orientation_Y_t-0 0.22 orientation_X_t-29 -0.96 linear_acceleration_Y_t-6 4.08 angular_velocity_Y_t-21 -0.02 linear_acceleration_Z_t-6 -6.98 linear_acceleration_Y_t-93 1.02 orientation_X_t-110 -0.96	 Feature Value linear_acceleration_X_t-0 0.93 orientation_X_t-0 -0.26 linear_acceleration_X_t-3 0.63 orientation_X_t-79 -0.25 angular_velocity_Z_t-0 -0.16 orientation_X_t-62 -0.25 orientation_X_t-110 -0.24 angular_velocity_Z_t-111 -0.15 orientation_W_t-100 -0.05 angular_velocity_Z_t-99 -0.18

Sensor Data Analysis

Experiment H

<table>
<tr><th>ModelNum</th><th>The training and validation loss for each training epoch</th><th>The training and validation accuracy for each training epoch</th></tr>
<tr><td>H-1</td><td>
</td><td>
</td></tr>
</table>

<table>
<tr><th>ModelNum</th><th>The heat map of the confusion matrix</th><th>The heat map of the normalized confusion matrix</th></tr>
<tr><td>H-1</td><td>
</td><td>
</td></tr>
</table>

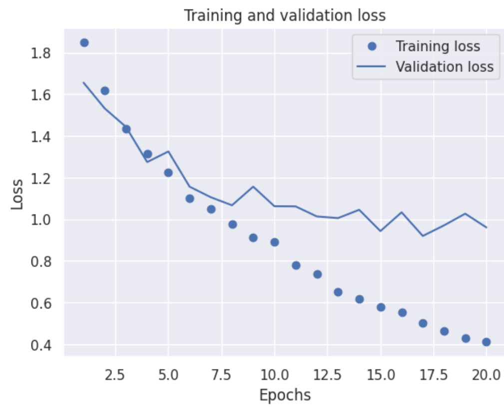
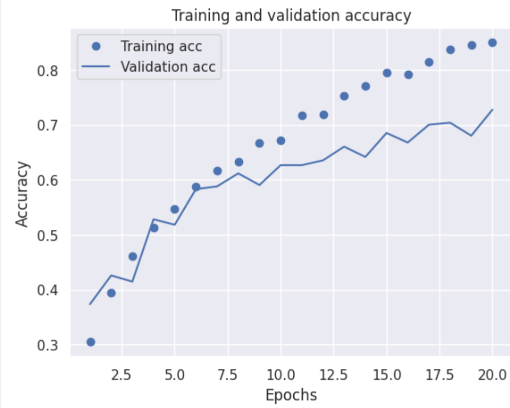
<table>
<tr><th>ModelNum</th><th>Activation value[0] for each class</th><th>t-SNE plot</th></tr>
<tr><td>H-1</td><td></td><td></td></tr>
</table>

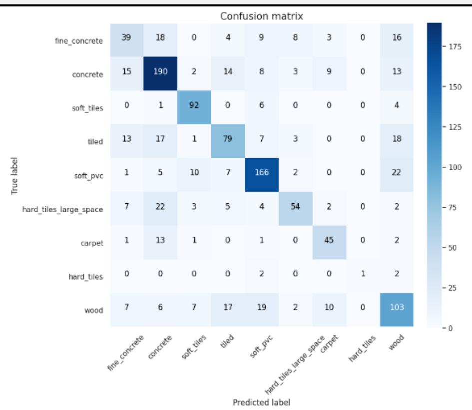
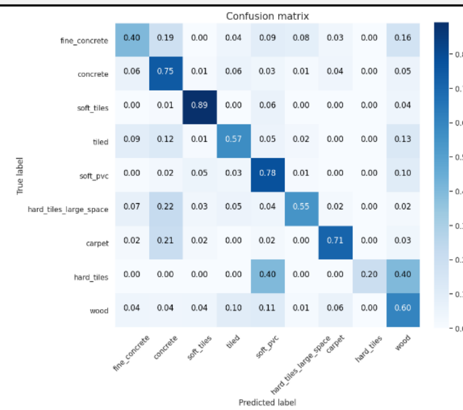
Sensor Data Analysis

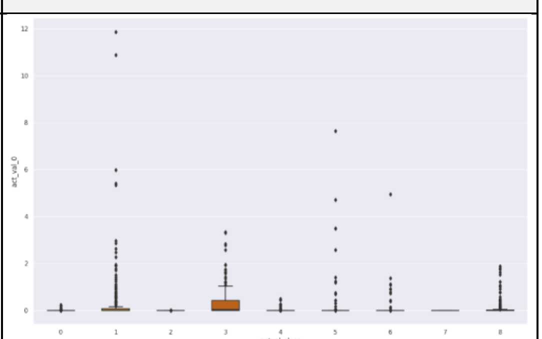
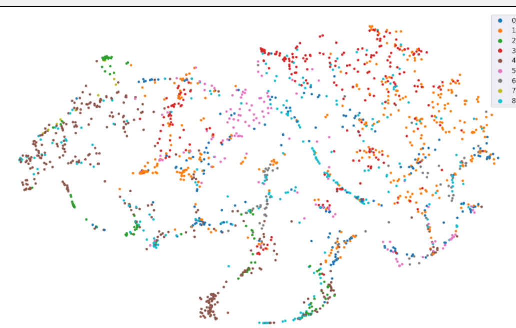
ModelNum	Lime explanation (prediction for series_id = 1) Actual class = 6 : carpet	Lime explanation (prediction for series_id = 4) Actual class = 4 : soft_pvc
H-1	Prediction probabilities 1 0.82 3 0.12 6 0.02 0 0.02 Other 0.01 Feature / Value orientation_X_t-0 -0.96 orientation_W_t-96 -0.14 orientation_W_t-8 -0.14 orientation_W_t-89 -0.14 angular_velocity_Z_t-30 -0.05 orientation_Y_t-41 0.22 orientation_Y_t-37 0.22 orientation_W_t-113 -0.14 orientation_W_t-100 -0.14 angular_velocity_X_t-33 -0.08	Prediction probabilities 4 0.61 3 0.28 1 0.04 0 0.03 Other 0.04 Feature / Value orientation_X_t-36 -0.25 orientation_X_t-51 -0.25 linear_acceleration_Z_t-8 -9.91 linear_acceleration_Y_t-57 2.23 linear_acceleration_Y_t-34 1.15 linear_acceleration_Z_t-24 -9.94 orientation_W_t-94 -0.05 orientation_Z_t-24 0.15 orientation_W_t-87 -0.05 orientation_X_t-5 -0.26

Sensor Data Analysis

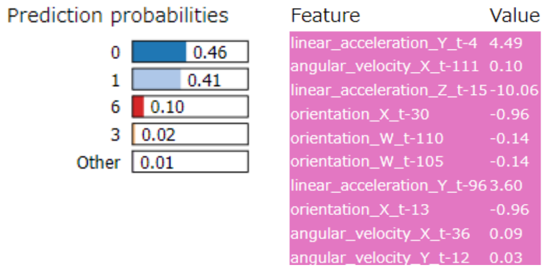
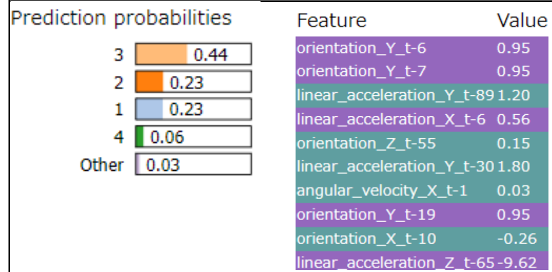
Experiment I

ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
I-1		

ModelNum	The heat map of the confusion matrix	The heat map of the normalized confusion matrix
I-1		

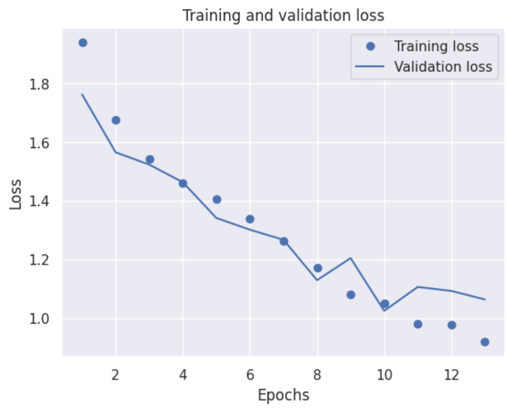
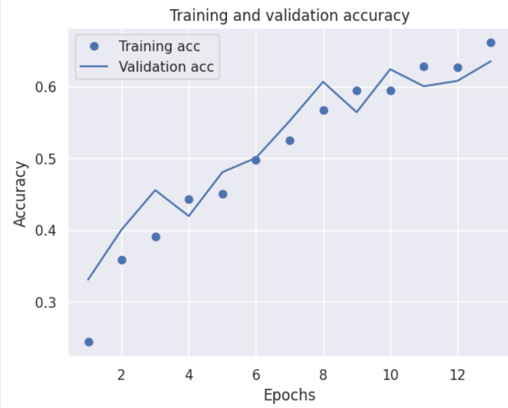
ModelNum	Activation value[0] for each class	t-SNE plot
I-1		

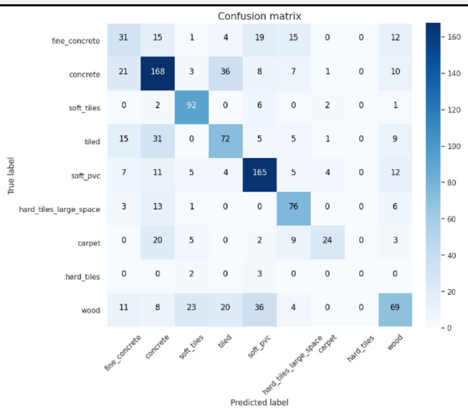
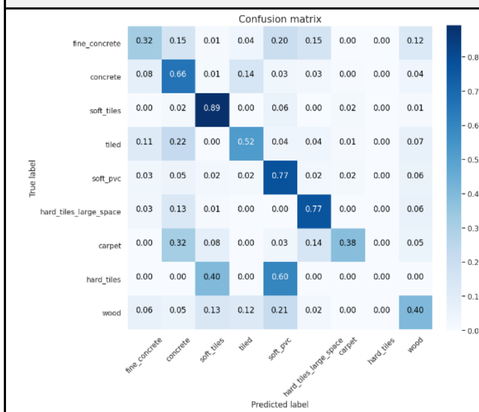
Sensor Data Analysis

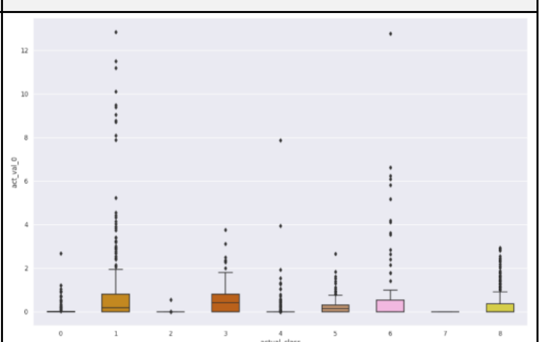
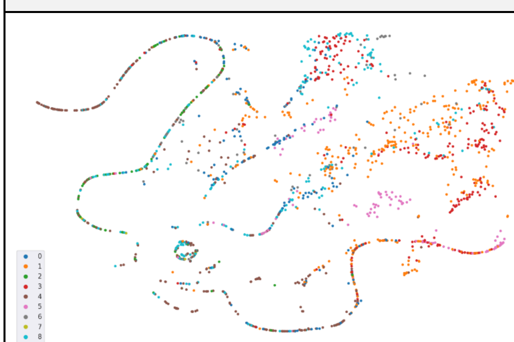
<table>
<tr><th>ModelNum</th><th>Lime explanation (prediction for series_id = 1)
Actual class = 6 : carpet</th><th>Lime explanation (prediction for series_id = 4)
Actual class = 4 : soft_pvc</th></tr>
<tr><td>I-1</td><td>
</td><td>
</td></tr>
</table>

Sensor Data Analysis

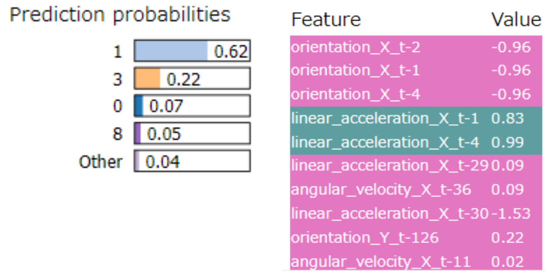
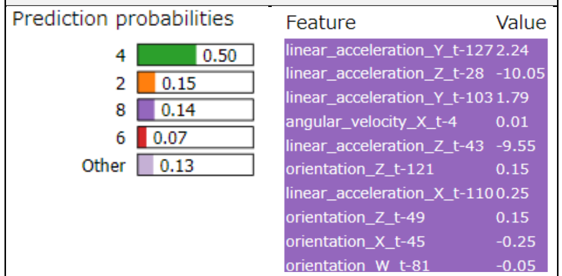
Experiment J

ModelNum	The training and validation loss for each training epoch	The training and validation accuracy for each training epoch
J-1		

ModelNum	The heat map of the confusion matrix	The heat map of the normalized confusion matrix
J-1		

ModelNum	Activation value[0] for each class	t-SNE plot
J-1		

Sensor Data Analysis

<table>
<tr><th>ModelNum</th><th>Lime explanation (prediction for series_id = 1)
Actual class = 6 : carpet</th><th>Lime explanation (prediction for series_id = 4)
Actual class = 4 : soft_pvc</th></tr>
<tr><td>J-1</td><td>
</td><td>
</td></tr>
</table>