Intelligent Sensor Fusion with Deep Learning Algorithm

Syed Muhammad Saim
Electronic Engineering
Hamm-Lippstadt University of Applied Sciences
Lippstadt, Germany
syed-muhammad.saim@stud.hshl.de

***Abstract*—Accuracy is a major concern on the basis of which efficient systems and equipments are designed. Intelligent Sensor Fusion, explained in this paper, is merged with Deep Learning so that union of these two concepts can be visualized. The basic implementation examples such as Camera Radar Fusion, Degradation of Ball Screws and Heterogeneous Human Activity Recognition with Dynamic Sensor Fusion are shown to not only clear the concept theoretically but also practically.**

I. Introduction

Inexorable developments require different branches of science and technology to merge with each other. A single field, when approach its peak, can still go further if combined with any other. Previously, we used to learn Electronics individually after which it merged with IT. Similarly, the field of Mechatronics is the combination of Mechanical and Electrical field. Among these, Sensor Fusion with Deep Learning is one such concept that is explained in this research work [2].

Intelligent Sensor Fusion and Deep Learning work hand-in-hand nowadays. New innovations are needed to be sensed as well as learned. For example, a humanoid robot has to sense any hurdle in front of him so that it can avoid. Furthermore, It also has to detect the temperature of the surroundings and act accordingly. Just like a human that have six senses with the help of which a human learns and trains. Eventually, a human make decisions after analyzing [3].

In this paper, sensor fusion is defined comprehensively. After this, Deep Learning approach is explained by giving a brief introduction of various sub topics which include CNN, RNN and TCN. These theoretical ideas are then implemented descriptively. Firstly implementation involves a general overview of a autonomous vehicle that how the LiDAR sensors and camera work hand-in-hand. The second implementation involves a ball screw example with a result in the form of graph. Finally, Heterogenous Human Activity is implemented with the help of accelerometer and gyroscope after which it has been coded with deep learning algorithm The graphs of which are also shown supported with a confusion matrix [7].

The main idea of this paper is to mention various aspects of applying the deep learning concepts with the help of sensor fusion. There are many such examples that are need to be studied. To summarize the idea, three of the core examples have been chosen to give a vivid view of the concept.

II. Sensor Fusion

A. Definition

Sensor fusion is the process of combining sensor data with data from a variety of sources to generate information with far less uncertainty than would be possible if these sources were used separately. Integrating data sources such as video cameras and WiFi localization signals, for example, could result in more exact interior object location prediction. The term uncertainty reduction could refer to the result of an evolving perspective, such as stereoscopic vision (depth information calculated by integrating two-dimensional images from two cameras at slightly different viewpoints), or it could refer to terms like more accurate, more complete, or more dependable [2].

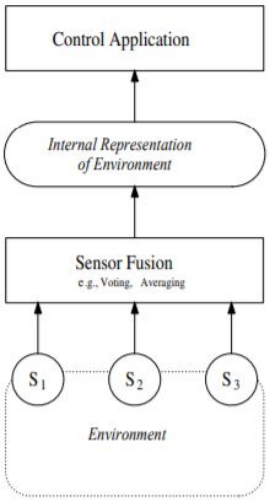

Figure 1. [2] Sensor Fusion Architecture

The data sources for a fusion process do not have to come from the same sensors. McKee divides fusion into three categories: direct, indirect, and fusion of the first two outputs. Direct fusion refers to the integration of sensor data from

a variety of heterogeneous or homogeneous sensors, soft sensors, and sensor data historical values, whereas indirect fusion refers to the use of preexisting knowledge about the environment and human input. As a result, sensor fusion is used to describe direct fusion systems, whereas information fusion is used to describe indirect fusion systems [2].

The above concept of sensor fusion does not need that inputs be provided by different sensors; it merely requires that sensor data or data derived from sensor data be combined. Sensor fusion systems, for example, are covered in the definition since they use a single sensor to take many measurements at different intervals and then combine them [2].

III. Deep Learning Approach

Deep learning is a subset of machine learning approaches in which algorithms are taught to learn how to construct complicated nonlinear systems, resulting in the refinement of computer systems through data. Deep Learning can be supervised or unsupervised. Supervised Deep Learning uses labeled training data for classification and regression tasks, while unsupervised Deep Learning uses unlabeled data to learn patterns or structures of the entire sample set for clustering, pattern recognition, or dimensionality reduction [3].

Figure 2 depicts the Deep Neural Networks (DNN) model structure, which contains an input layer, many hidden layers, and an output layer. Hidden layers in image data frequently learn simple elements like corners or contours, which when integrated in a deep structure form a rather complicated model. These layers are made up of artificial neurons and are joined by edges in the middle. Weights are normally assigned to the edges and neurons, with the goal of learning them during the training phase. Learning the weights is the process of modifying weights while the system sees signals being processed [3].

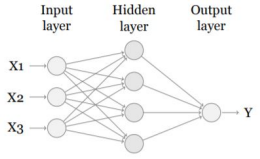

Figure 2. [3] Artificial Neural Network

Weight changes are made by minimizing a cost function, which is complete when adding more observations no longer reduces the error rate. As a result, it's possible that the error rate will never be zero. Once the error has been discovered, the idea is to back-propagate the information through the network by computing the gradient of the cost function. As a consequence, each gradient component reveals how much each weight is affected by the inaccuracy. Back-propagation is known for vanishing (and sometimes exploding) gradients, which occur as gradients approach 0 and prevent weight updates. As a result, the network will either cease to be able to learn or will learn at a glacial pace [3].

When analyzing network results, it's vital to make sure that the overall generalization is correct. The model is neither underfitted nor overfitted because the purpose is to make reasonable predictions on unknown data. When a model is overfit, the weights have been overly tailored to the training data and are unable to perform adequately when faced with unknown input. An overfitted model is identified by a large discrepancy between training and validation error. When underfitting a model, on the other hand, the model may have been further trained for weight adjustments. Underfitting occurs when the model cannot achieve a low enough training error [6].

Regularization strategies have gained a lot of attention because there is a large risk of overfitting the model while building DNN algorithms. Dropping out turns off network components that aren't in use. Early stopping is a method of monitoring the training process and terminating it when the validation loss has not decreased for several epochs. Data augmentation artificially improves the diversity of input data by inserting noise and/or bias. Models that use these regularization techniques are more resistant to uncertain data [6].

A. Useful Network Architecture for the Problem Domain

The reader will learn about the many types of network designs that are acceptable for the issue domain in this part. Three forms of network design are of interest: recurrent neural networks, convolutional neural networks, and temporal convolutional networks [3].

Recurrent Neural Networks (RNN) were initially described in 1986 and are a sort of neural network. They're utilized to deal with sequential data and let previous network outputs influence future forecasts. As demonstrated in Figure 2.2, during the training phase, remembering what has been learned from prior inputs while producing new outputs. The x0:t inputs are given to network A, which generates h0:t predictions. The parameters of A are shared across the sequence, allowing the model to be expanded and applied to a larger set of data before making a forecast [3].

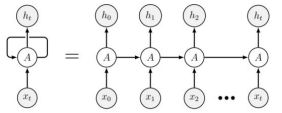

Figure 3. [3] Recurrent Neural Network

As a result of the sequential approach, processing is slow, accessing information from prior time steps is difficult, gathering the entire context may be too onerous, and vanishing gradient problems may occur. The latter is due

to the fact that long-term dependencies have exponentially increasing or decreasing multiplicative gradients. Many sequential-based methods use Long Short Term Memory (LSTM) to solve difficulties with vanishing gradients that occur frequently during the back-propagation step when training RNNs [3].

In 1997, the LSTM was invented to assure a continuous flow of mistakes. The LSTM better captures long-term interdependence by using a memory cell to store states across time. In addition to the memory cell, the LSTM unit has an input, output, and forget gate. The three gates regulate the flow of information into and out of the cell. The LSTM unit cell stores back-propagation defects until the units learn to forget them. As a result, learning long-term dependencies with higher context is easier than with RNN. As a result, LSTM networks must be trained on massive volumes of data, resulting in a longer training period and increased hardware memory requirements for the network [3].

Convolutional Neural Networks (CNN) are a form of deep neural network that was created to learn features and classify data. One of the earliest fully established algorithms announced in 1989 was used for a pattern recognition task. A CNN model is frequently built in a hierarchical manner, with the purpose of collecting basic data into increasingly sophisticated characteristics. As a result, they don't need as much pre-processing as other classification algorithms because they can learn these qualities with enough practice. In addition to pattern recognition, modern CNNs are used for supervised classification problems [3].

CNN has sparse interactions (small relevant characteristics can be identified), is translation equivariant (output and input translate in the same way), and kernels (filters) share weights. Because several neurons can share weights and biases, the model only needs to learn one set of parameters, which saves memory. A model typically includes an input layer, as well as multiple convolutional, sub-sampling, and fully connected layers [1].

Raw data is generally fed into the input layer of a CNN, which in the case of IMU measurements will be $ninput$ 6 1, resulting in a 1-D grid of time-series data [3].

By extracting features, the convolutional layer builds feature maps. Convolutional layers are commonly used to apply padding, stride, and dilation. The stride parameter indicates the distance between two convolutions where the kernel should be applied spatially. Padding can be classified into two types: valid padding, which decreases the input dimension to the kernel size, and same padding, which increases or preserves the input dimension [3].

The neurons of a convolutional layer process data only from its receptive field, which is a small fraction of the previous layer. Because there are so many levels of convolutions, each neuron in a layer can take input from a larger area than the layer before it. The dilatation thinned out the receptive field while keeping the kernel size the same. As a result, when illustrated in Figure 2.3, the receptive field widens as the dilation factor is increased in multiple layers [1].

Local averaging is used in the sub-sampling layer to reduce the resolution of the feature map and the output's sensitivity to shifts and distortions. A pooling layer can be used instead of the subsampling layer, however it is not generally trainable. Maximum/average pooling, which takes the kernel's maximum and average values, is a frequent technique. When using sub-sampling layers, fewer parameters must be stored, reducing the model's memory needs and the number of operations necessary to compute the output [3].

The job of the completely linked layer is to connect every neuron in the system from one layer to the next before performing the final classification [3].

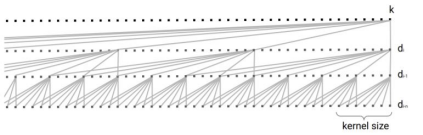

Figure 4. [3] Varying dilation factor per layer $d_i = 1, 4, 16$.

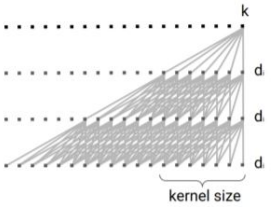

Figure 5. [3] Dilation factor equal on each layer $d_i = 1, 1, 1$.

Temporal Convolutional Networks (TCN) are a sort of convolutional neural network that has just been constructed. TCN's purpose is to make sequence modeling capable of dealing with long-term temporal relationships. TCN is made up of 1D convolution layers that are dilated and causal, with equal input and output lengths. Causal convolutions are network outputs that are unaffected by future inputs. The network becomes adaptable for sequential data as a result of dilated and causal dilations, with no leakage from future to past, thanks to its temporally and spatially large receptive fields. TCN has the advantage of not being sequentially dependent on network predictions, allowing parallelization across GPU cores and hence rapid training [3].

Text synthesis and speech recognition have both been investigated using TCN from a generative and discriminative standpoint. In a widely cited academic paper, a generative model for Raw Audio based on dilated convolutions is proposed. The network topology based on dilated convolutions is known for its ability to demonstrate extraordinarily large receptive fields while requiring minimal data sets. Large IMU

data receptive fields can be utilized to forecast states based on a longer chain of historical measurements, not just the most recent but also those from the past. As a result, if the data of interest is thought to be substantially related to time, broad receptive fields are desired [1].

IV. Implementation

A. Radar and Camera Fusion Architecture For Object Detection

This method improves the object detection accuracy of existing 2D object detection networks by combining camera data and projected sparse radar data in the network layers. The radar directly obtains information about object distance and radial velocity. Figure 1: Van can locate things in a two-dimensional plane parallel to the ground even when the lens is obliterated by a water droplet. The radar sensor, unlike the camera, is unable to gather height information. We create a network architecture that handles data from both cameras and radar sensors [5].

RetinaNet and a VGG backbone are used to build the neural network, as illustrated in. To accommodate the additional radar channels in the enhanced image, the network has been enlarged. A 2D regression of bounding box coordinates as well as a classification score for the bounding box are the network's outputs. As proposed in, the network is trained with concentrated loss. Throughout the first convolutional layers, our baseline approach employs a VGG feature extractor [5].

The amount of data contained in a single radar return differs from that contained in a single pixel. The distance of an item to the ego-vehicle as sensed by the radar can be considered more essential to the driving task than the basic color value of a camera pixel. If both sensors are fused by concatenation in an early fusion, we should assume that the varied data are semantically equivalent. Because we cannot strongly substantiate this assumption, the fusion of the first layer of the network may not be desirable. At the deeper levels of the neural network, the input data is compressed into a denser representation, which ideally contains all of the critical input information. We design the network in such a way that it learns which depth level of data fusion is most helpful to overall loss minimization because the abstraction level of the information provided by each of the two sensor types is difficult to quantify. The high-level structure of the network is shown in Figure 6. The VGG blocks make up the fusion network's core pipeline, which is seen in the graph's center branch. The camera and radar data are merged and fed into the network on the top row. In this branch of the network, the VGG layers analyze both the camera and radar data. Raw radar data is also sent into the network at higher layers of the network in the left branch via max-pooling at appropriately scaled input sizes. The radar data is concatenated with the output of the preceding fused network layers of the main branch of the network. The blocks P3 through P7 correspond to the Feature Pyramid Network (FPN) proposed in, in which the radar channels are added at each level and fused by concatenation. The bounding box regression and classification blocks are used to process the outputs of the FPN blocks. The optimizer indirectly trains the network at which depth levels the radar data is fused with the maximum impact by altering the weights to the radar characteristics at different layers [5].

For camera and radar data, we present a new multi-modal sensor fusion training technique. The Dropout technique is used in this strategy. We disable all input neurons for the camera image data, rather than single neurons, for random training steps. This is done at a rate of 0.2 of all training photos. BlackIn is the name of this technique. Network dropout at the final layer inspired BlackOut. The network is forced to rely more on radar data due to a paucity of camera input data. The goal is to teach the network how to comprehend the information value of sparse radar data without relying on the much denser camera representation [5].

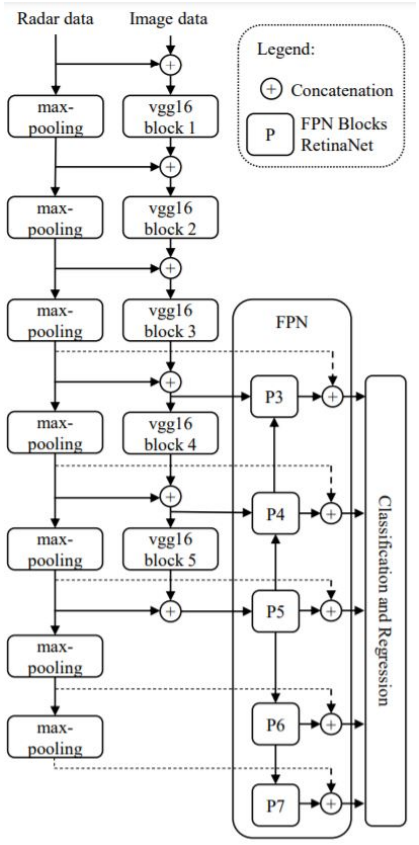

Figure 6. [5] Structure of CameraRadarFusionNet (CRFNet)

B. Degradation Monitoring of Ball Screws

Because the ball screw has a complicated structure and a wide distribution range, a single signal recorded by a single sensor is difficult to fully and precisely convey its status. In the proposed deep learning-based multi-sensor data fusion method, parallel superposition on frequency spectra of signals is directly done, and deep belief networks

(DBN) are established by using fused data to adaptively mine available fault characteristics and automatically identify the degradation condition of ball screw. The test is meant to capture vibration signals from a ball screw in seven distinct deterioration circumstances using five separate acceleration sensors. Finally, multi-sensor data fusion using a neural network is used to monitor the degree of degradation [7].

As shown in Fig. 7, the proposed method consists of four steps: data preprocessing, unsupervised pre-training, supervised training, and degradation condition monitoring [7].

In this study, the FFZD4010R-3 type of ball screw was mounted on the acceleration performance degradation test bench to investigate its deterioration characteristics. This test bench is made up of one drive motor, two slide guides, three ball screws, four rack and pinion units, and five magnetic power brakes, as shown in Figure 8 [7].

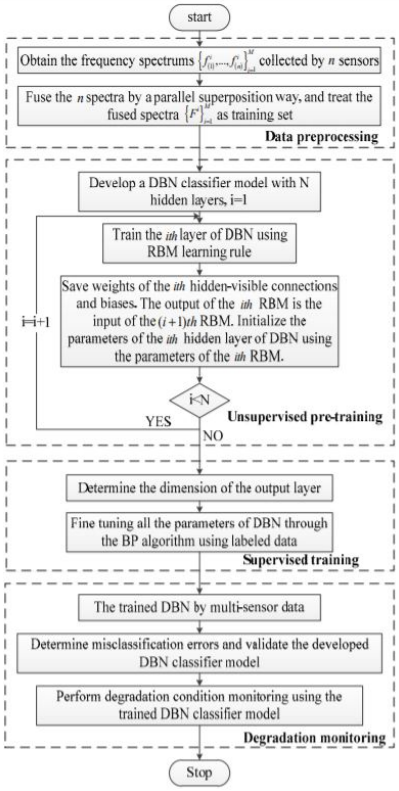

Figure 7. [7] Flowchart for the proposed method.

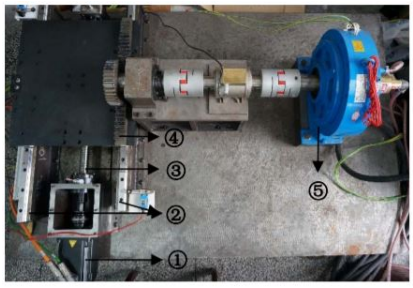

Figure 8. [7] Degradation Test Bench for Ball Screw.

Vibration signals are captured using 5 acceleration sensors in 7 deterioration grades and 9 operational situations. The cross combination of three speeds (100r/min, 300r/min, 800r/min) and three axial loads (0kN, 1kN, 2kN) results in nine working situations. 5 acceleration sensors mounted in various places and directions collect five groups of vibration signals. Ball nut and two bearing seats are among the positions, and screw directions include axial and radial. At the same time, five acceleration sensors collect data. In each degrading condition, each acceleration sensor gathers 120 samples for the first 6 working conditions and 90 samples for the last 3 working conditions. Hence Each deterioration condition collects 4950 samples, and each acceleration sensor collects 6930 samples for a total of 7 degradation grades. Finally, the ball screw deterioration vibration signal has 34650 samples collected by 5 sensors in various positions and directions under 9 working situations and 7 degradation grades [7].

The obtained degradation data must next be preprocessed. Each sample's spectrum, which is a 512-length vector, is calculated separately. As a result, 5 groups of signals recorded by 5 sensors form 5 spectrum matrices, each with a line number of 6930 and a column number of 512. Then, using parallel superposition, combine the 5 spectrum matrices to create a spectrum matrix of 6930 lines and 2560 columns. As a result, the number and dimension of input data are 6930 and 2560, respectively, and the fused spectrum matrix is used as DBN's input. When the ball screw is in the nth degradation grade, it is represented as a seven-dimensional vector with the nth value equal to 1 and the other values equal to 0 [7].

The trained DBN classifier model contains a bottom data layer, a top output layer, and two hidden layers of 500 and 100 neurons, respectively. The input layer's neuron number is equal to the input spectrum dataset's dimension, and the output layer's neuron number is likewise equal to the number of defined ball screw degradation grades. In this investigation, the DBN structure is 2560-500-100-7. In this RBM training process and back propagation learning process, the maximum number of training epochs is set to 50 and 600, respectively, the learning rate is 0.1, and the batch size is 63. DBN's weights are started at random, and biases are set to zero. 70 percent of the data samples in the trial are chosen at random for training, while the rest are used for testing. Twenty times the trial was repeated [7].

Figure 5 shows the categorization results for degradation conditions using the proposed technique. These twenty trials have an average training accuracy of 99.06 percent and an average testing accuracy of 91.89 percent. Validation results show that the suggested method is successful and that these seven ball screw deterioration scenarios can be discriminated with high accuracy. DBN is capable of learning complex nonlinear relationships between input data sets and distinct degradation states, which accounts for its excellent accuracy

[7].

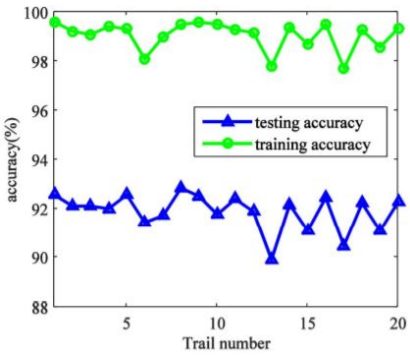

Figure 9. [7] Classification Results using proposed method.

C. Hetrogeneous Human Activity Recognition with Dynamic Sensor Fusion using Deep Learning Models

Human activity is among the most difficult concepts to apply in the real world. We can employ a combination of sensors from smartphone and wearable devices to address this heterogeneity. We used deep learning techniques and constructed two models with Feed Forward Fully Connected Network and Recurrent Neural Network adaptable frameworks to reduce inconsistency and improve accuracy. We use the data from two sensors - the accelerometer and gyroscope on a smartphone and smartwatch - to train the models. The outcome indicates the applicability of our model to existing human recognition algorithms, as well as its advantages over existing approaches [4].

UCI's machine learning repository has contributed the datasets. The dataset was created to benchmark human activity recognition algorithms (classification, automatic data segmentation, sensor fusion, feature extraction, and so on) in real-world contexts; specifically, the dataset was gathered with a variety of different device models and use-scenarios, in order to reflect sensing heterogeneities that would be expected in practical deployments. An accelerometer has three axes: two for most two-dimensional movement and a third for three-dimensional location. Most smartphones employ three-axis models, but autos only use a two-axis model to calculate the moment of impact. These devices have a high sensitivity since they are designed to detect even the tiniest changes in acceleration. The accelerometer's sensitivity determines how easily it can measure acceleration. A gyroscope detects angular velocity using a three-axis acceleration sensor (X, Y, and Z). It is made up of three double T structural pieces that react to shifts and movements. Magnetometer, GPS, proximity sensor, and other suitable sensors can be employed in the suggested model [4].

The network accepts dynamic information as input, which travels via the input layers and out the output layer. Each node on the adjacent levels is connected to every other node, analogous to how neurons in the human brain are connected. The following are the network's optimal hyper parameters: Number of hidden layers - 4, Neurons - 100, Epochs - 0 to 50, Learning Rate - 0.001 [4].

The hidden layer has a linear function A = cx that receives an input and multiplies it with random weights. It is activated by the ReLu activation function A(x) = max(0,x), which returns x if x is positive and 0 otherwise [4].

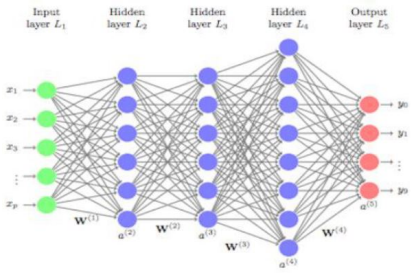

Figure 10. [4] Feed Forward Neural Network.

Both the gyroscope and accelerometer data gave extrapolated with accuracy upto 71 percent. The percentage accuracy on trained as well as test data are shown while tuning optimal hyper parameters. The best result was obtained for ReLu activation with aforementioned hyper parameters [4].

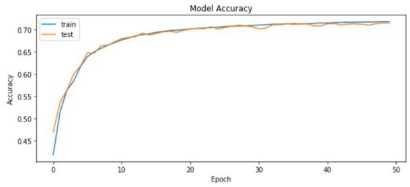

Figure 11. [4] Model Accuracy.

During the model's training procedure, we were able to reduce the model loss from 1.8 percent to 0.8 percent [4].

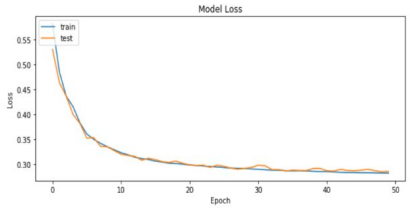

Figure 12. [4] Model Loss.

We developed a confusion matrix to compare the model evaluation of the input data versus the projected data, and the model delivered an 80 percent prediction on four of the six objectives supplied into it to better understand the performance of the proposed model [4].

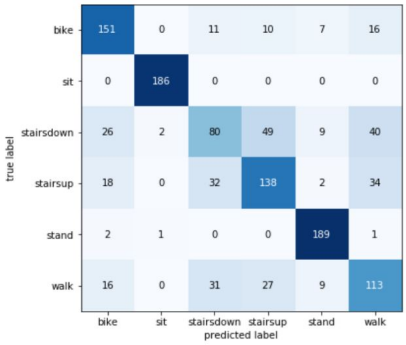

Figure 13. [4] Confusion Matrix for FNN.

V. Conclusion

Implementing various examples in this paper gives us the idea that of Sensor fusion can be a great assistant for Deep Learning. Normally a dataset has to be prepared that can be fed into deep learning implementation. With the help of Sensors fusion technique, both the concepts can be applied simultaneously. Reliable data can be collected from different sensors and hence, can be learned and trained at the same time.

References

[1] Adit Deshpande. The 9 deep learning papers you need to know about (understanding cnns part 3). *adeshpande3. github. io. Retrieved*, pages 12–04, 2018.

[2] Wilfried Elmenreich. *Sensor fusion in time-triggered systems*. PhD thesis, 2002.

[3] Shaun Michael Howard. *Deep learning for sensor fusion*. PhD thesis, Case Western Reserve University, 2017.

[4] Sakorn Mekruksavanich, Anuchit Jitpattanakul, and Patcharapan Thongkum. Placement effect of motion sensors for human activity recognition using lstm network. In *2021 Joint International Conference on Digital Arts, Media and Technology with ECTI Northern Section Conference on Electrical, Electronics, Computer and Telecommunication Engineering*, pages 273–276, 2021.

[5] Felix Nobis, Maximilian Geisslinger, Markus Weber, Johannes Betz, and Markus Lienkamp. A deep learning-based radar and camera sensor fusion architecture for object detection. In *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–7. IEEE, 2019.

[6] Vadim Romanuke. Appropriate number and allocation of relus in convolutional neural networks. *Research Bulletin of the National Technical University of Ukraine" Kyiv Politechnic Institute"*, (1):69–78, 2017.

[7] Li Zhang and Hongli Gao. A deep learning-based multi-sensor data fusion method for degradation monitoring of ball screws. In *2016 Prognostics and System Health Management Conference (PHM-Chengdu)*, pages 1–6. IEEE, 2016.