Improve the LSTM and GRU model for small training data by wavelet transformation

1st Jengnan Tzeng*
Dept. of Mathematical Sciences,
National Cheng-Chi University
Taipei, Taiwan
https://orcid.org/0000-0003-4072-7274

2nd Yen-Ru Lai
Dept. of Mathematical Sciences,
National Cheng-Chi University
Taipei, Taiwan
freedom870601@gmail.com

3rd Ming-Lai Lin
Dept. of Mathematical Sciences,
National Cheng-Chi University
Taipei, Taiwan
f861231@gmail.com

4th Yu-Han Lin
Dept. of Mathematical Sciences,
National Cheng-Chi University
Taipei, Taiwan
harry02261112@gmail.com

5th Yu-Cheng Shih
Dept. of Mathematical Sciences,
National Cheng-Chi University
Taipei, Taiwan
leo978810@gmail.com

***Abstract*—Regarding collision prediction technology, the most common are car reversing radar and infrared rays, which provide warnings by sensing the distance between objects and cars. Although radar detection is accurate, radar systems that can provide instant 360-degree feedback are very expensive. The cost of infrared is much lower, but they cannot be applied to high-temperature environments. As the result, the technology of preventing collisions using only images from cameras is an important artificial intelligence topic in recent years. If a low-resolution CCD and artificial intelligence technology can be used to achieve a certain degree of accuracy in collision prediction, then low-cost anti-collision technology is worth looking forward to. Furthermore, we hope to provide anti-collision warnings on motorcycles and bicycles using this technology. In order to achieve this goal, computational simplification is a technical threshold. It's only when simple calculations achieve high-precision prediction can we meet the low power consumption requirements for image AI to be applied small vehicles. Therefore, we hope to find out a better image representation basis and combine it with AI technology to fulfill the requirements of less calculation and high accuracy. In addition, we also hope to create models with sufficient accuracy with small training data. This experiment will reduce development costs and get better efficiency in the early stage of developing ADAS.**

***Index Terms*—wavelet, Haar basis, AI, ADAS, small training data**

I. Introduction

In order to make it possible for the computer to determine whether there is a collision by only inputting the image, and as the image is continuously inputted in a time series method, we accordingly choose an artificial intelligence network method that can process time-series data. In this case, the famous recurrent neural network (RNN) method meets this characteristic [1], as this method is mainly used to process sequential input, especially when inputs are of various lengths [2]. We can define each observation data as a time series sequence and for each sequence we give it its label. A RNN training model contains the following three network layers: the first layer is the input layer, the second is the hidden layer and the third is an output layer. And the hidden layer is a recursive structure, and the training coefficient at the current time t is generated by the current input integrated with its previous training coefficient in the hidden layer. If we look into the inner structure of the hidden layer, we can see that it is constructed by many small neural networks, which are connected with one another and sharing their updated training coefficients. Such network in the hidden layer is capable of memorizing, and therefore, it is a great tool for time serial data AI training [3].

However, the RNN structure has the problem of gradient vanishing, so a new method is proposed. We call it a long short term memory network (LSTM) [4]. It has three control units in the hidden layer, which are the input gate, the forget gate, and the output gate. Through the training data, we see these three gates can be selectively open or closed judged by a sigmoid function. In the input gate, open means receiving data and closed means objecting. If the data are received, the newly generated memory unit will join into the long-term memory. In the forget gate, open means memorizing the previous nodes and closed means forgetting the previous nodes. In the output gate, open means the current result is added to the output and closed means no output .

Gated Recurrent Unit (GRU)was proposed by Cho 2014 [5]. It is also designed to solve thevanishing gradient problemwhich comes with a standard recurrent neural network, but it has more simple structure than LSTM. GRU is considered as a variation of LSTM and in some cases produces equally excellent results. It has gating units that modulate the flow of information inside the unit without having separate memory cells.

In the practical application of these artificial intelligence networks, the input data can be an intuitive data or a transformed data. Just as the Fourier representation is often applied

Funded by the Ministry of Science and Technology (project number MOST 107-2115-M-004-002)

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

to sound signals [6], wavelet is an important method for representing image data. Wavelet representation mainly focuses on orthogonality and multi-resolution representation. Unlike the Fourier transformation, the support of the wavelet function is compact. Thus, wavelet can capture signal information in both frequency and physical domain at the same time. The simplest Haar wavelet can use the average and difference of two data to make a multi-resolution data representation structure. Ingrid Daubechies proposed an orthogonal wavelet in 1990 [7] that there are general forms of orthogonal wavelet, and then the academic community has made a more in-depth exploration of wavelet knowledge. However, in imitating the information transmission of biological vision or hearing system, we believe that if the biological information system is a kind of wavelet transformation, it would not be the fancy Daubechies's high order wavelet transformation, it is would be the Haar wavelet.

In the survey, we have tried two methods of collision. One method developed by Jagannath Aghav in 2017 uses deep learning to extract the relative speed of moving objects towards a body to determine whether a collision will occur [8]. The other method is a neural network work that imitates locust vision [9]. Notice that the learning materials required in these two methods need to be sufficient, which is different from our article since we discuss how to obtain preliminary learning results with a small number of learning samples. As such, when applying these two methods to these small samples collected, we cannot get a stable model with learning effect, i.e., , the prediction accuracy is not obviously more than 50Therefore, this article will compare the difference in prediction accuracy between the LSTM and GRU methods with and without Haar wavelet transformation.

II. Mathematical notation

We first define the mathematical notations that will be used in this paper. Let $V_k = (F_{k,1}, F_{k,2}, \ldots, F_{i,T})$ be the input video, where $F_{k,t}$ is the t-th still picture in the k-th input video. We assume that each $F_{k,t}$ is an m-by-n matrix and elements of $F_{k,t}$ are integer number that refers to the gray level of grayscale image.

For each $F_{k,t}$, there are m rows and n columns. Each row or column in the matrix can be regarded as a one-dimensional discrete data. Haar function $\phi(x)$ is a step function. Assume that $\phi(x)$ is defined by

$$\phi(x) = \begin{cases} 1 & \text{if } x \in [0,1], \\ 0 & \text{o.w.} \end{cases} \tag{1}$$

It is very special that $\phi(x)$ can be reconstructed by the combination of its scaling and shifting functions. For example,

$$\phi(x) = \phi(2x) + \phi(2x-1). \tag{2}$$

This function is called a scaling function or father wavelet. Let $\psi(x) = \phi(2x) - \phi(2x-1)$ be the mother wavelet. We can see that $\phi(2x) = (\phi(x) + \psi(x))/2$ and $\phi(2x-1) = (\phi(x) - \psi(x))/2$. If a space V_j is spanned by $\phi(2^j x - k)$ for $k \in Z$ and $j \in Z^+$, then $V_j \subset V_{j+1}$. We called this is a multi-resolution structure. We also define a space W_j is spanned by $\psi(2^j x - k)$ for $k \in Z$ and we can easily see that $V_{j+1} = V_j + W_j$.

The 1-D discrete wavelet transformation is starting from a given sequence a_k for $k = 0, \ldots, 2N-1$. We can consider the function $f(x) = \sum_{k=0}^{2N-1} a_k \phi(2^j x - k) \in V_j$. Since $V_j = V_{j-1} + W_{j-1}$, we can represent $f(x)$ by $\phi(2^{j-1}x - k)$ and $\psi(2^{j-1}x - k)$. That is $f(x) = \sum_{k=0}^{N-1} \mathcal{L}_k \phi(2^{j-1}x - k) + \mathcal{H}_k \psi(2^{j-1}x - k)$, where $\mathcal{L}_k = (a_{2k-1} + a_{2k})/2$ and $\mathcal{H}_k = (a_{2k-1} - a_{2k})/2$. We can only focus on a sequence a_k with length $2N$ be decomposed to two sequences $\mathcal{L}_k$ and $\mathcal{H}_k$ with length N. The sequence $\mathcal{L}_k$ is the low pass coefficient and $\mathcal{H}_k$ is called the high pass coefficient. Convert discrete sequence a_k to the low pass sequence $\mathcal{L}_k$ and high pass sequence $\mathcal{H}_k$ is the first layer 1-D wavelet transformation. We can apply the wavelet transformation only to the low pass part $\mathcal{L}_k$ to get the second layer wavelet representation. Moreover, we can continuously repeat this process until the sequence is represented almost by mother wavelets.

For the first 2-D wavelet transformation, we apply each row of the input matrix the first layer 1-D wavelet transformation and put the low pass part to the left (L part) and the high pass part to the right (H part). See figure 1 (a) to (b). And then we apply each column the first layer 1-D wavelet transformation again and put the low pass part to the top (LL part and HL part) and the high pass part to the bottom (LH part and HH part). See figure 1 (b) to (c). Just as the 1-D wavelet transformation, the 2^{nd} layer wavelet transformation is applied to the 2-D wavelet transformation only on the LL part. See Figure 1 (c) to (d). We can continuously this process to make almost all components coefficients of the mother wavelets.

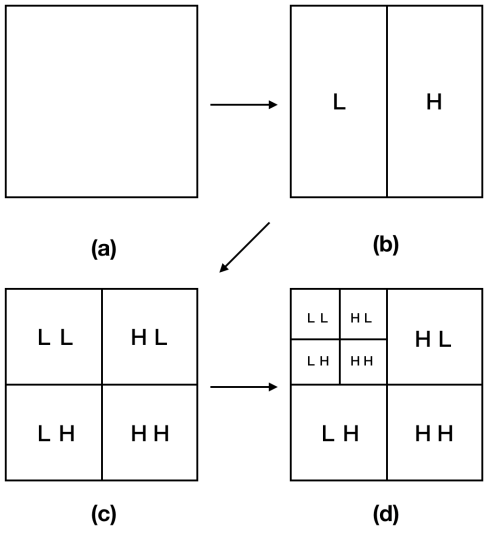


Fig. 1: Example of 2^{nd} layer 2-D wavelet transformation.

We then define Φ_k as the k-layer 2-D Haar wavelet transformation. For example, $X_1 = \Phi_1(X_0)$, where X_0 is a 2^{ℓ_1}-by-2^{ℓ_2} matrix. Then the upper left corner of X_1 is the LL pass of X_0, where $X_1(i,j) = (X_0(2i-1, 2j-1) + X_0(2i-1, 2j) + X_0(2i, 2j-1) + X_0(2i, 2j))/4$ for $i \leq 2^{\ell_1 - 1}$,

$j \le 2^{\ell_2-1}$; the upper right corner of X_1 is the HL pass of X_0, where $X_1(i,j) = (-X_0(2i-1,2j-1) + X_0(2i-1,2j) - X_0(2i,2j-1) + X_0(2i,2j))/4$ for $i \le 2^{\ell_1-1}$, $j \ge 2^{\ell_2-1}$; the bottom left corner of X_1 is the LH pass of X_0, where $X_1(i,j) = (-X_0(2i-1,2j-1) - X_0(2i-1,2j) + X_0(2i,2j-1) + X_0(2i,2j))/4$ for $i \ge 2^{\ell_1-1}$, $j \le 2^{\ell_2-1}$; and the bottom right corner of X_1 is the HH pass of X_0, where $X_1(i,j) = (X_0(2i-1,2j-1) - X_0(2i-1,2j) - X_0(2i,2j-1) + X_0(2i,2j))/4$ for $i \ge 2^{\ell_1-1}$, $j \ge 2^{\ell_2-1}$. Then, the general $X_k = \Phi_k(X_0)$ can be defined as below:

$$X_k(i,j) = \begin{cases} Y(i,j) & \text{if } i \le 2^{\ell_1-k+1} \text{ and } j \le 2^{\ell_2-k+1} \\ X_{k-1}(i,j) & \text{o.w.} \end{cases} \tag{3}$$

where $Y = \Phi_1(X_{k-1}(1:2^{\ell_1-k+1}, 1:2^{\ell_2-k+1}))$.

Given a fixed number of layer τ, we define

$$W_n = (\Phi_\tau(F_{n,1}), \Phi_\tau(F_{n,2}), \ldots, \Phi_\tau(F_{n,T})). \tag{4}$$

That is V_n is the n-th video input with standard representation and W_n is the n-th video input with τ-th layer Haar wavelet representation. We also define $\tilde{W}_n$ as below:

$$\tilde{W}_n = (\Phi^H_\tau(F_{n,1}), \Phi^H_\tau(F_{n,2}), \ldots, \Phi^H_\tau(F_{n,T})), \tag{5}$$

where $\Phi^H_\tau(F_{n,t})$ captures only the HH parts of $\Phi_\tau(F_{n,j})$, and then resizes these HH part matrixes to vectors and combines them to one long vector. For each n, the level vector $Y_n = (y_{n,1}, y_{n,2}, \ldots, y_{n,T})$ where $y_{n,t}$ is the sub-level of each $F_{n,t}$. If V_n or W_n is the case of collision, then $y_{n,t}$ is increasing and we set $y_{n,1} = 0$ and $y_{n,T} = 1$; else $y_{n,t} = 0$ for all $t = 1, \ldots, T$.

Next, we briefly introduce the notations and settings we used in the AI learning process. The following x_t will be replaced by $\Phi^H_\tau(F_{n,t})$ if we are looking for the AI model with wavelet transformation or F_i if without wavelet transformation. The general RNN updates the recurrent hidden state h_t by

$$h_t = \begin{cases} 0, & t = 0 \\ x_t \psi(h_{t-1}, x_t), & \text{o.w.,} \end{cases} \tag{6}$$

where ψ is a nonlinear function, such as the composition of a logistic sigmoid with an affine transformation. To update the recurrent hidden state is often represented as

$$h_t = g(Ax_t + Bh_{t-1}), \tag{7}$$

where g is a smooth, bounded function, such as a logistic sigmoid function or a hyperbolic tangent function.

In the LSTM model, we retain a memory unit at time t, say c^j_t. The formula of output h^j_t can be

$$h^j_t = o^j_t tanh(c^j_t), \tag{8}$$

where o^j_t is determined by the amount of the previous memory content exposure. This output gate is of the following form:

$$o^j_t = \sigma(A_o x_t + B_o h_{t-1} + D_o c_t)^j, \tag{9}$$

where σ is a logistic sigmoid function and D_o is a diagonal matrix. The memory cell c^j_t is computed by partially forgetting the existing memory c^j_{t-1} and adding a new memory $\tilde{c}^j_t$, where $c^j_t = f^j_t c^j_{t-1} + i^j_t \tilde{c}^j_t$ and $\tilde{c}^j_t = tanh(A_c x_t + B_c h_{t-1})^j$. The design of c^j_t includes the forget gate f^j_t and the input gate i^j_t. These gates are formulated as

$$\begin{aligned} f^j_t &= \sigma(A_f x_t + B_f h_{t-1} + D_f c_{t-1})^j, \\ i^j_t &= \sigma(A_i x_t + B_i h_{t-1} + D_i c_{t-1})^j, \end{aligned} \tag{10}$$

where the subindex f and i refer to the forget gate and input gate and D_f, D_i are diagonal.

The GRU is similar to LSTM and its structure is simple. The relative formula are the following:

$$h^j_t = (1 - z^j_t) h^j_{t-1} + z^j_t \tilde{h}^j_t, \tag{11}$$

where an updated gate z^j_t refers to the amount of the updates and

$$z^j_t = \sigma(A_z x_t + B_z h_{t-1})^j. \tag{12}$$

The candidate activation $\tilde{h}^j_t$ is computed by

$$\tilde{h}^j_t = tanh(Ax_t + B(r_t \cdot h_{t-1}))^j, \tag{13}$$

where r_t is a reset gate and the operator '$\cdot$' is an element-wise multiplication. And the reset gate r^j_t is computed by

$$r^j_t = \sigma(A_r x_t + B_r h_{t-1})^j. \tag{14}$$

That's all for notation setting.

III. Methodology and Experimental results

We use Python 3.5 as our programming language with Jupyter notebook IDLE interface. The using packages are pywt, keras, sklearn and pywt. The keras and sklearn packages are often used in machine learning and the pywt package is used in the discrete wavelet transformation.

Given that most of data about collision experiments in Open data is the data which obtain collision images from the perspective of a third party, we hence do not use the data in Open data, say, the 32 crash datasets on data.world website, for experiments. We use a rolling table tennis ball to hit the camera to simulate the process of a vehicle collision. A fixed camera is placed on one end of the table, and on the other end a table tennis ball is rolled by a person toward the camera. To control the speed of the ball, we make a slope track. Due to the slope is fixed, as we release the ball from the higher end of the track, we will get a similar speed of the moment the ball leaves the track in our 200 repeated actions. By adjusting the direction of the track, we can control whether the ball will hit the camera. Due to the human operation, the result of the speed each time will be slightly different. We will collect two data types. One is data with collision, and the other is data without collision. Each video data is composed of 7-20 frames, and the size of each image frame is 2048 * 2048. We take the last 7 frames as training materials. When a collision occurs, we set the last frame to be the moment when the collision occurs. When there is no collision, we set the last frame to be the moment when the ball passes by the camera. In total, we have collected 100 collision data and 100 collision-free data. Each input contains 7 time-series photos and each photo is stored in a grayscale format.

We use a standard 2-D Haar wavelet transformation to process our input image. We also use a three-layer Haar wavelet transformation to transfer data from a standard basis to the Haar wavelet basis. Then, the pre-input data looks like $W_n = (\phi_3(F_1), \phi_3(F_2), \ldots, \phi_3(F_7))$. For each $\phi_3(F_t)$, we extract the HH parts and resize them into the vector type and combine them to a long vector. Therefore, we obtain the input data $\tilde{W}_n$, where $\tilde{W}_n = (\phi_3^H(F_1), \phi_3^H(F_2), \ldots, \phi_3^H(F_7))$. For fair comparison, we rescale the input original frames to a 443-by-443 matrix. (Notice that if we do not rescale the input original frames to the small size, the performance of the AI model without wavelet transformation will become worse than the resized performance.) This total matrix size is similar to the three HH parts of each element of $\tilde{W}_n$. $\tilde{V}_n$ represents these resized input data. The numbers of $\tilde{W}_n$ and $\tilde{V}_n$ are both 100. We will separate these data into the training and testing sets. Then we compare the performance of LSTM and GRU models.

For the training LSTM and GRU model, we design 5 layers of network. They are kernel layer, recurrent kernel layer, bias layer, dense layer and second bias layer. To obtain the better performance, we use an orthogonal matrix as the initial matrix in both LSTM and GRU, which will also make a better convergence rate. The initial orthogonal matrix is determined randomly by the Python packages.

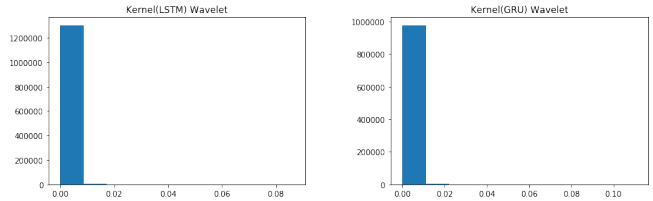
Because different inputs will obtain different outputs, we use K fold cross validation to avoid the overfitting. In our experiments, we set $K = 10$. That is we separate our data into two parts, each time we use 20 data for testings and 180 data for trainings. When the input changes, the output of each layer will be changed too. We randomly split the data to training sets and testing sets, and repeat the splitting 64 times. From each training set, we build an AI model to examine the prediction ability of each testing set. We will also observe the distribution of each component of the output matrix for each layer, and then to find out the important components. If some standard deviation of the component is small and the value is significant, then we will keep this component. On the other hand, if the relative standard deviation is large and the value is small, we will forcibly set the component becomes zero. Finally, we will retrain the network to see whether the model can perform well.

For each K fold cross validation, we construct four models. The first is LSTM model with wavelet data ($\tilde{W}_i$ as input), the second is LSTM model with original data ($\tilde{V}_i$ as input), the third is GRU model with wavelet data, and the final is GRU model with original data. With these four models, we can analyze whether the Haar wavelet transformation will improve the performance.

Table I shows the average accuracy of the testing set in LSTM and GRU model with/without wavelet transformation. Each number is the result of the above-mentioned 64 operations. We can see that the LSTM model is better than the GRU model, no matter we use wavelet transformation or not. Haar wavelet representation actually improves the average accuracy in both LSTM and GRU model, especially in LSTM model. In GRU model, wavelet transformation elevates 7.8752% accuracy from 49.687% to 53.60%; and it elevates very much in LSTM model from 49.99% to 90.0%. In the training sets, all the accuracy rates are 100%. Notice that without wavelet transformation, both LSTM and GRU model do not complete the AI learning process, because they guess all results to be none collision.

TABLE I: Comparison of the LSTM and GRU model in testing set with/without wavelet transformation.

Case	Average accuracy	variance
LSTM (with wavelet)	0.900	0.007
GRU (with wavelet)	0.536	0.010
LSTM (without wavelet)	0.4999	0.011875
GRU (without wavelet)	0.49687	0.0092

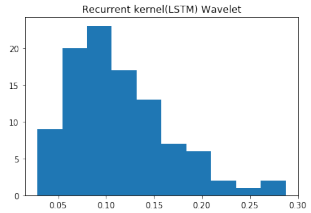


(a) LSTM (b) GRU

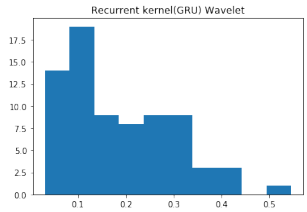
Fig. 2: Distribution of the standard deviation of Kernel layer

As each model is determined by the training set, we would like to observe the stability of the models. Hence, we observe the distribution of the standard deviation of individual elements of the matrix corresponding to the kernel layer, the recurrent kernel layer, and the density layer. Figure 2 has two graphs, which are the standard deviation distributions of the LSTM model and the GRU model. We can see from the distribution map that most of the standard deviations are concentrated near 0. This indicates that the values of most matrix elements are nearly the same in different models, which means these AI models are stable.

From figures 3 and 4, we can observe that the matrix elements corresponding to the recurrent kernel layer and dense layer are more sensitive to the input data. The sensitivity of the dense layer is greater than that of the kernel layer. Since the kernel layer is more stable than the recurrent kernel layer and dense layer, we try to force the elements close to 0 to be 0. This approach is an attempt to eliminate unimportant elements of the kernel layer so that we can predict with less data. We know that the input data can be greatly improved after being converted by wavelet transformation. When we force non-significant elements to be 0, we use partial elements to calculate the interpretation result, so the accuracy usually decreases. Due to the large amount of data in the kernel layer, it is necessary to appropriately reduce its size while maintaining a certain degree of prediction accuracy. We take a uniform grids on the interval $[0.005, 0.05]$ as the threshold value, and set the absolute value of the matrix element of the kernel layer less than the threshold value to be 0. After setting

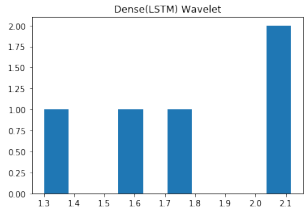


(a) LSTM

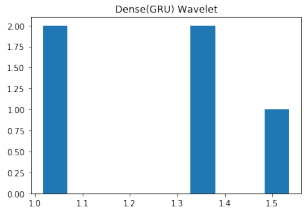


(b) GRU

Fig. 3: Distribution of the standard deviation of Recurrent Kernel layer



(a) LSTM



(b) GRU

Fig. 4: Distribution of the standard deviation of Dense layer

up this kernel layer, we continued to use training data to learn the parameters of other layers in the model. Lastly, we observe the performance of the revised model on accuracy.

Figure 5 shows the predicted ability of the model when we forced elements of the kernel layer less than a threshold value to be 0. The Y axis of figure 5 is accuracy, and the X axis is the percentage of kernel layer that we forcibly set to be 0. As the percentage is increasing, the accuracy will gradually decrease as expected. We observe that even if the percentage is as high as 85%, that is, we only use 15% of the memory capacity for the model, our accuracy is still more than 70%. Notice that the result demonstrated in Figure 5 is based on the best model among all models we train according to our experience, but the best model is very likely to be different depending on various factors in the training process.

In previous experiments, we used a fixed slope to control the speed of the table tennis ball rolling. We are curious about what the model learned with a fixed speed would look like if it is applied to a sample with a various speed. So we recorded another data set that includes 60 non-collided images and 60 collided images. We can see that on the speed-control samples, we have 90% accuracy, which will quickly drop to 63.33% when we use the non-speed-control samples. The decrease in accuracy shows that our model is really poor at resolving

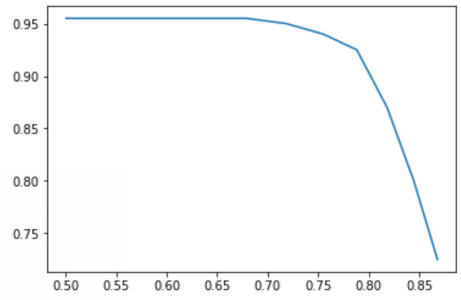


Fig. 5: Relationship between the accuracy and model memory.

samples with speed changes. However, this 63.33% accuracy is still better than 49.99% of LSTM model without wavelet transformation.

We have another way to present the wavelet basis indeed helps visual AI recognition. We sort the values of the matrix elements corresponding to the kernel layer from small to large. We cut 1000 equal parts between the maximum and minimum values. In each small interval, we count how many elements in the matrix fall within this interval. Then we convert these accumulated numbers into probabilities.

We can roughly treat these 1000 intervals as 1,000 possible information representations, and calculate their information entropy through the probability just calculated. In the case of LSTM model with wavelet transformation, the average number of matrix element values in the kernel layer is $-4.5265064 \times 10^{-5}$, the standard deviation is 0.04842207, the median is $-4.194437497 \times 10^{-5}$, the quartile of Q_1 is $-2.93498808 \times 10^{-2}$, and Q_3 is 2.928800×10^{-2}. Most of the data is centered around zero. After our calculation the entropy is 5.3574225293. In the case of LSTM model without wavelet transformation that has the best accuracy rate among others, the average number of matrix element values in the kernel layer is 1.9123004×10^{-6}, the standard deviation is 0.0022573355, the median is $1.4606289369 \times 10^{-6}$, the quartile of Q_1 is $-1.52123105 \times 10^{-3}$, and Q_3 is $1.52504939 \times 10^{-3}$. Most of the data is centered around zero. After our calculation the entropy is 8.3288037021. The maximum information entropy in these 1,000 message types is 9.965784284662087. We see that the message entropy 8.3288037021 of the case LSTM without wavelet transformation is close to the completely random number case. Hence, applying a wavelet transformation really helps the model to capture important information, rather than 50% to 50% random guessing. This outcome is also reflected in our ability to predict whether it is a collision image.

IV. Conclusion

Most of the AI image research indicates that investigating collisions problem require more than 6,000 data. We try to complete the training of AI collision models with fewer data. By using image data with high-high-frequency parts of Haar wavelet expansion as input data, we observe that although only 200 sample data is used as AI learning, our prediction accuracy

has been greatly improved to 90%. When using new data that does not control speed to make predictions, although the accuracy of the prediction has dropped significantly, the model using wavelet transformation still shows higher accuracy after comparing with the model without wavelet transformation. Therefore, we verified that the wavelet basis really has the ability to improve AI image recognition.

We have seen that Haar wavelet transformation is a transformation that requires very few computational cost, and hence it can enhance the AI image recognition for small sample. For training the AI image models with larger sample, we therefore suggest that the Haar basis should be used to exhibit the image signal inputted. From the angle of entropy, we save significant memory space in computation and obtain fairly good recognition rate, which makes it possible to apply the ADAS technique to motorcycles or even bicycles. In the future, we need to expand the training data to areas like the marching and collision motions of bicycles in the real world so that we can develop a more complete ADAS AI model.

ACKNOWLEDGMENT

My thanks to Ming-Lai Lin, Yu-Han Lin, and Yu-Cheng Shih, who helped collect the image data of the collision, and thank Yen-Ru Lai for completing most of the AI programming and calculation. We are also deeply grateful to the Ministry of Science and Technology (project number MOST 107-2115-M-004-002) for supporting our experiments.

REFERENCES

[1] H. Li, B.S. Manjunath and S.K. Mitra, "Graphical models and image processing," Elsevier,1995
[2] A. Graves, "Sequence Transduction with Recurrent Neural Net-works, CoRR, vol. abs/1211.3711, 2012.
[3] M. Bod en, "A guide to recurrent neural networks and backpropagation," 2001.
[4] Malhotra, P., Vig, L., Shroff, G., & Agarwal, P., "Long short term memory networks for anomaly detection in time series," Presses universitaires de Louvain, 2015
[5] Chung, J., Gulcehre, C., Cho, K., & Bengio, Y, "Empirical evaluation of gated recurrent neural networks on sequence modeling," arXiv preprint arXiv:1412.3555. 2014
[6] R. G. Osuna, "Introduction to Speech Processing, 1st ed. Computer Science & Engineering- Texas A & M University, 2016.
[7] I Daubechies, "Ten Lectures on Wavelets,SIAM, 1992
[8] Aghav, J., Hirwe, P., & Nene, M., "Deep Learning for Real Time Collision Detection and Avoidance, In Proceedings of International Conference on Communication, Computing and Networking, 2017
[9] Yue, S. and Rind, F. C., "Collision detection in complex dynamic scenes using and lgmd-based visual neural network with feature enhancement, Neural Netw. IEEE Trans. 17, 705-716. doi:10.1109/TNN.2006.873286