MATLAB-BASED APPROACH TO INVESTIGATING DATASET TESTING AND TRAINING FOR ENHANCED HUMAN-LIKE FRAME PREDICTION USING CONVOLUTIONAL NEURAL NETWORKS IN DIVERSE SCENES THROUGH DEEP LEARNING

Carlo N. Romero.[1]
[1]Our Lady of Fatima University, Quezon City Philippines

Corresponding Email: cnromero@fatima.edu.ph

Available Online: November 2023
Revised: October 2023
Accepted: September 2023
Received: October 2022

Volume I Issue 4 (2023)
DOI: 10.5281/zenodo.10299557
E-ISSN: 2984-7184
P-ISSN: 2984-7176
https://getinternational.org/research/

Abstract

Deep Learning has been applied to train the convolutional neural networks (CNNs) for accurate frame prediction. Using CNNs, a MATLAB-Based approach is used to investigate dataset testing and training techniques for obtaining enhanced human-like frame prediction using CNNs in diverse scenes. Furthermore, studies were explored in Deep Learning, which is a kind of Machine Learning that can be trained, supervised, semi-supervised and unsupervised. Specifically, the proposed study applies deep learning methods, including Convolutional Neural Networks (CNNs), for next-frame prediction. The Catz Dataset is utilized as the training data for this investigation. The experimental results reveal that CNNs can indeed be used to achieve human-like frame prediction in diverse scenes. The best performing model, a hybrid CNN and LSTM network, exhibits a significantly improved perceptual distance rating of 26.7127, outperforming the initial CNN model. These findings demonstrate the potential of CNNs trained using deep learning techniques for accurate frame prediction tasks. The study has also shown that impact of the training and testing ratios on the performance of an enhanced human-like frame prediction using CNNs and MATLAB. The experiments through MATLAB have shown that higher training percentage means that a larger portion of dataset for training the model have been used while a lower training percentage shows that a large portion of the dataset reserved for testing the model's performance.

Keywords: *Frame Prediction, Convolutional Neural Networks, Long Short-Term Memory Networks, Computer Vision, Deep Learning, Artificial Intelligence, MATLAB-Based Approach*

Recommended Citation: Romero.1Carlo N. (2023). MATLAB-BASED APPROACH TO INVESTIGATING DATASET TESTING AND TRAINING FOR ENHANCED HUMAN-LIKE FRAME PREDICTION USING CONVOLUTIONAL NEURAL NETWORKS IN DIVERSE SCENES THROUGH DEEP LEARNING. GUILD OF EDUCATORS IN TESOL INTERNATIONAL RESEARCH JOURNAL, 1(4), 10–20.
https://doi.org/10.5281/zenodo.10299557

INTRODUCTION

In recent years machine learning has been applied to many fields. One such kind of machine learning, Deep Learning, has been applied to a vast array of image processing problems.

Predicting motions and frames using Artificial Intelligence is definitely possible. With the use of a training model and data sets that can train it to become better at generating images, these things aren't even fantasy anymore. With this possibility, it can be possible for a learning model to estimate motion using two images or predict motion using just one image.

Next-Frame Prediction is predicting something that happens next to a particular scene or image, predicting it in the form of an image or a few images. (Zhou, Dong, Saddik, 2018). It is one of the more popular fields of Computer Vision and Deep Learning. This kind of prediction is based on the understanding of the information in past images, those that have already occurred so far. The input of the network is the previous few frames, and prediction is/are the next frame. The predictions can not only predict Human motions, but background, object, and the motion of other living beings. Modeling Contents and dynamics from videos or images is the main task for next-frame prediction, which makes it different from motion prediction.

Computer Vision is a facet of Deep Learning that aims to give vision to computers and robots. This study has given rise to things such as self-driving cars. Frame Prediction can improve this in many ways, for example, having the computer predict if it will collide with a hazard, so as to avoid being hit. This all can be achieved by training Convolutional Neural Networks to model special data and have a Recurrent Neural Network work in tandem with the CNN to solve Time sequence problems that CNNs are not equipped to work on.

In Figure 1, the researcher decided to test CNNs and RNNs along with an RNN called Long Short-Term Memory Network, which is by far one of the most capable RNNs for making predictions using Time Data.

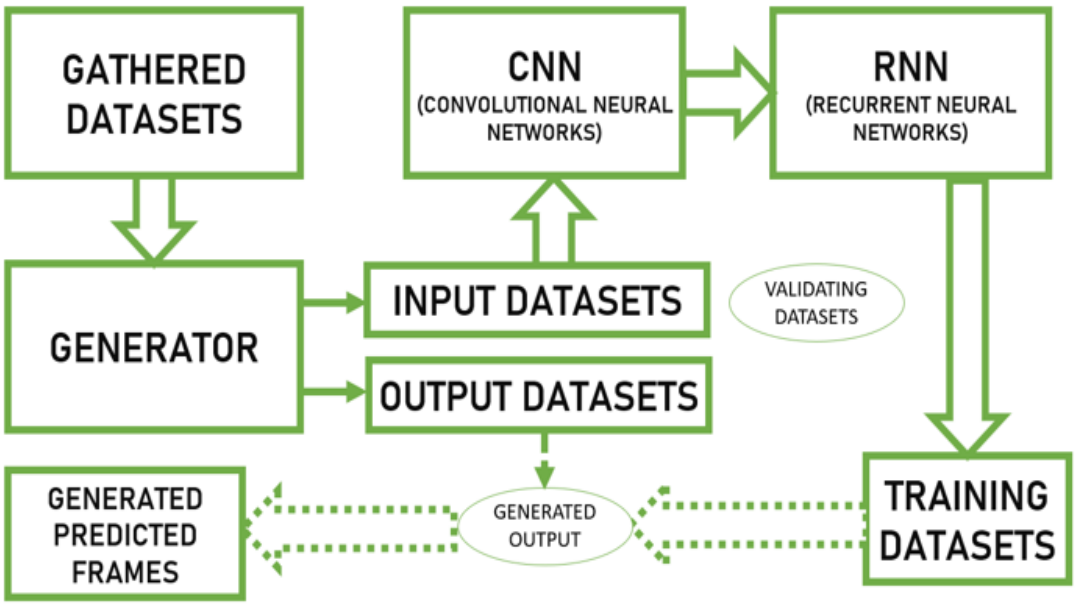

Figure 1: Conceptual Framework

Objectives

This research investigated the effects of different dataset testing and training methods on the performance of convolutional neural networks (CNNs) for human-like frame prediction in diverse scenes.

Specifically, this study aimed to answer the following questions:

1. What are methods to develop a MATLAB-based framework for testing and training CNNs for human-like frame prediction?
2. How to identify the best dataset testing and training methods for achieving enhanced human-like frame prediction in diverse scenes?
3. What are the ways to generate a predicted image that uses five frames of a video and compare the predicted frame to the actual final frame?

METHODS

Project Design

The research will train an Artificial Intelligence (AI) Model Dataset and predict the next frame of an image or video frame and provide an original frame and predicted frame output using the following methods:

Long Short-Term Memory Networks: The researcher used an LSTMs so that we can have a short-term memory where the Model will look at 5 pictures.

Convolutional Neural Networks: The researcher trained a Convolutional Neural Network that outputs predicted frames from the Catz Dataset. It first gets 5 sample frames to train with and then outputs in result, the predicted frame alongside the first frame, or the input.

Google Colab: The researcher used the Google Colab platform so as not to limit the operational speed of the computer, utilizing google colab's cloud GPUs.

Keras: The researchers would have used Tensorflow as well, but they use an AMD Graphics card. AMD Graphics cards sometimes have problems with Tensorflow and would require too many workarounds.

Tensorflow: The researchers however still need tensorflow to be able to work as well, although making Keras the main API of choosing.

Project and Development

The Figure 2 shows the Phases that the system undergoes in order to create its output. The first phase in essentially preparing the data to undergo prediction. The Generator gets five images from the Catz Dataset and a result image, for training. It then labels the five images as Input, and then returns to the image log. The second phase is where Data Augmentation and the Training and validation algorithms are prepared along with the required Neural Networks. The third phase is quite possible the longest depending on Epochs and Batch Size. It is a 6 Layer training model, where the inputs are continuously rerun in order to get better and better predictions as the training goes on. The number of epochs from Base CNN to Functional Style is set to 10 while the last third of the training includes more epochs, meaning more steps and longer training. Finally, the fourth phase in which the Training and Validation Data is

stored along with Input and Output Images. The Five Images from before are labeled as Input and the Predicted Images are labeled output. The First frame of the original image is on the left and the predicted frame is on the right.

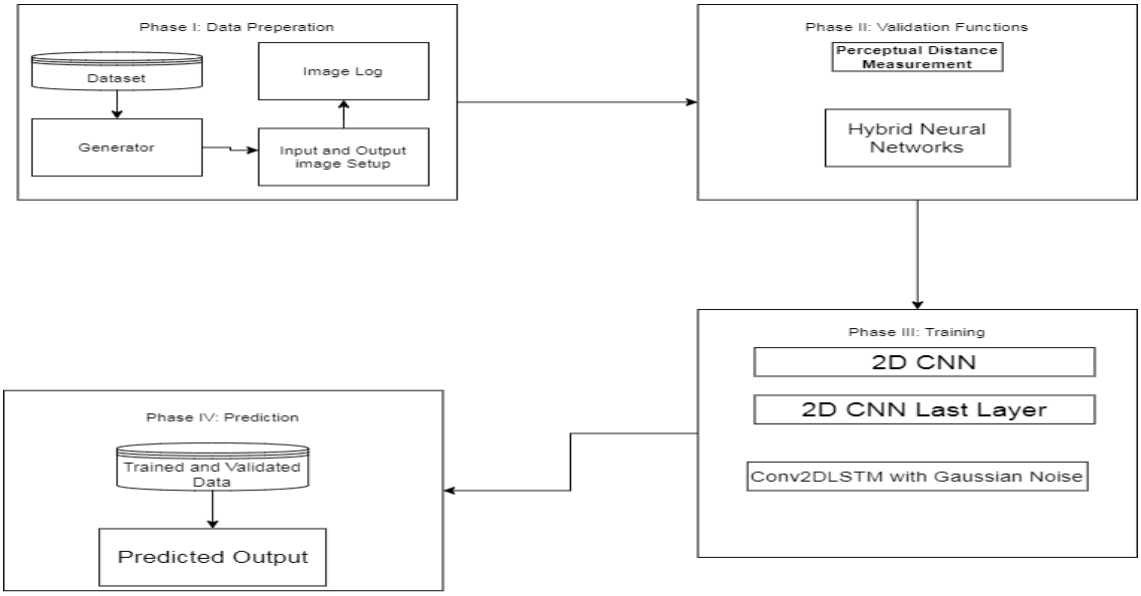

Figure 2: Phases of the System

MATLAB Codes (Testing and Training)

```
% Perform frame prediction using the trained network on videos

% Load or capture a test video
testVideo = VideoReader('path_to_test_video.mp4');

% Create a VideoWriter object for the output video
outputVideo = VideoWriter('output_video.avi', 'Motion JPEG AVI');
outputVideo.FrameRate = testVideo.FrameRate;
open(outputVideo);

% Read and process each frame of the test video
while hasFrame(testVideo)
    frame = readFrame(testVideo);

    % Preprocess the frame
    frame = imresize(frame, imageSize(1:2));
    frame = double(frame) / 255; % Normalize the frame to [0, 1]

    % Perform frame prediction
    predictedFrame = predict(net, frame);

    % Convert the predicted frame to uint8 format
    predictedFrame = uint8(predictedFrame * 255);

    % Write the predicted frame to the output video
    writeVideo(outputVideo, predictedFrame);
end

% Close the output video file
close(outputVideo);
```

Figure 3: Testing and Training Codes using MATLAB

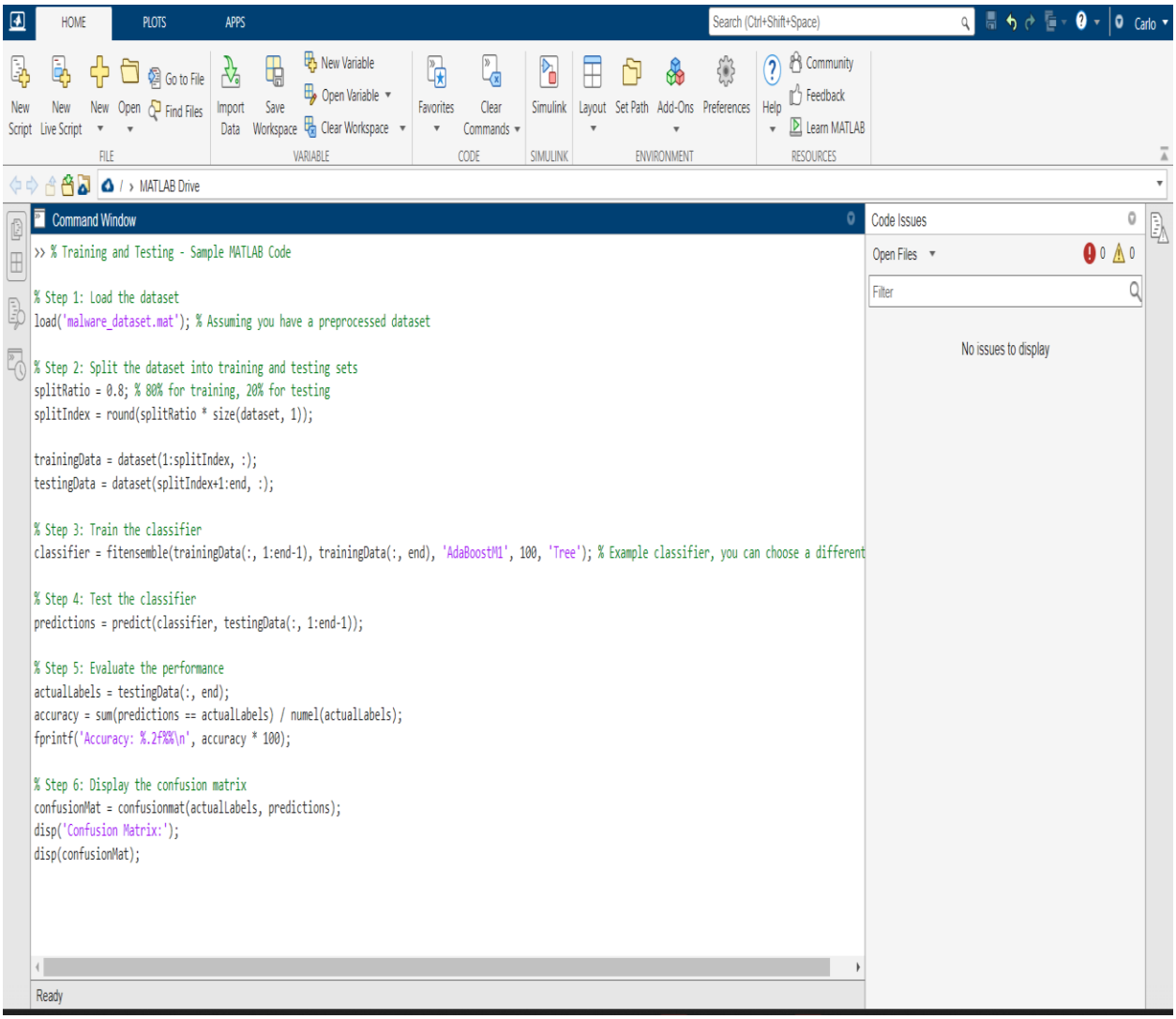

Figure 4: Phases of MATLAB Programming

In the proposed research, the researcher first load and preprocess the dataset using an imageDatastore, then split it into training and testing sets. We define a CNN architecture, specify training options, and train the network using the trainNetwork function, then for frame prediction on images, you need to load or capture a test image, preprocess it, and use the trained network to predict the frame. The predicted frame is then displayed.

RESULTS and DISCUSSION

This chapter presents the result and findings of the research report. The accuracy and performance of the Artificial Intelligence Model is also presented in this chapter with the use of the Convolutional Neural Network.

The researcher also used a Generator to feed into the fit function. A Data Generator essentially gives the researchers the ability to make modifications to the data as each data passes. The researchers used the Keras Image Data Generator framework. What this does is, instead of creating a variable with all the data, it has a function that will

return batches of training data examples. Basically, it allows for stretching, squeezing, rotating, and enlargement of the images, which also comes with the benefit of being able to reduce memory consumption and make training faster either by, reducing the resolution of an image, or making the image smaller.

Figure 5: System Phases

The first CNN Model used by the researchers is a 2D Convolution on the inputs from the dataset. The input is as presented previously, 96,96, 15 with the output being 3 frames. The first model is going to take each frame and it will do a 2D Convolution across all 15 frames and with a 3 frame output.

The code for this particular model will be presented here as well as the training procedure:

```
#2D Convolutional Model
wandb.init(config=hyperparams)
config = wandb.config
model = Sequential()
model.add(Conv2D(3, (3, 3), activation='relu', padding='same', input_shape=(config.height, config.width, 5 * 3)))
model.compile(optimizer='adam', loss='mse', metrics=[perceptual_distance])
model.fit_generator(my_generator(config.batch_size, train_dir),
                    steps_per_epoch=steps_per_epoch//4,
                    epochs=config.num_epochs, callbacks=[
    ImageCallback(), WandbCallback()],
    validation_steps=validation_steps//4,
    validation_data=my_generator(config.batch_size, val_dir))
```

This is the code for the 2D Convolutional Model. The Kernel is set to 3,3. The padding is set to the same for all inputs and outputs. Input shape will be defined by the hyperparameters Then compile the model, set optimizer to adam, loss to mse, and metrics to the perceptual distance function so it returns it.

Then call fit generator, which calls on the earlier generator for which takes generators that return tensors as input, in which case the earlier mentioned generator will be passed on to this code.

Then run the code and now the training procedure begins.

Table 1. Training Data the first CNN model

Epoch	Loss	Perceptual Distance	Val_Loss	Val_Perceptual Distance
1	0.0476	115.7747	0.0266	82.8251
2	0.0239	76.4415	0.0206	72.1625
3	0.0186	64.8026	0.0112	58.8218
4	0.0152	58.3414	0.0142	50.6381
5	0.0124	51.9734	0.0077	49.3901
….				
45	0.0058	29.8100	0.0021	26.2738
46	0.0052	27.4291	0.0051	27.1419
47	0.0049	26.5542	0.0056	26.6047
48	0.0065	30.4695	0.0132	31.3303
49	0.0053	27.4971	0.0043	26.8302
50	0.0059	29.0719	0.0082	30.6704
….				
95	0.0054	27.1183	0.0034	27.0859
96	0.0053	26.1695	0.0109	24.3569
97	0.0047	24.8629	0.0025	22.8550
98	0.0053	26.6195	0.0094	26.8313
99	0.0050	25.2611	0.0036	23.8095
100	0.0058	27.6982	0.0036	28.9328

This table shows the training procedure that the learning model is undergoing. It shows how much epochs are left before finishing. This table is shows the first to hundredth epochs and the training data per epoch. These are the inputs and outputs after the model completes all 100 steps are completed.

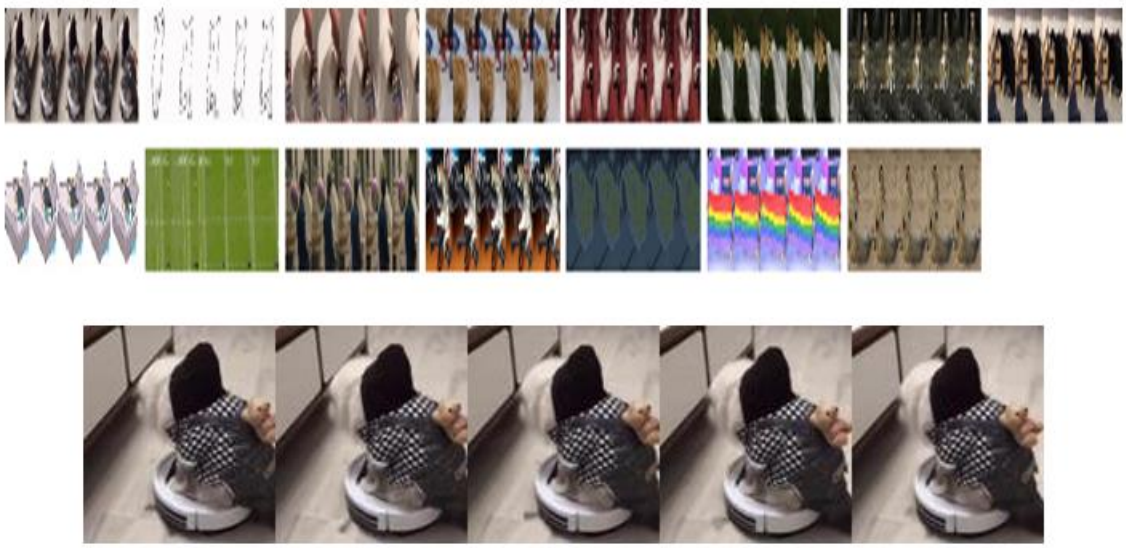

Figure 6: Input of the First Model

This Figure 6 shows the inputs from the Catz Dataset for the first model. There are actually five frames shown per input. The Final frame of all the inputs will be the output for the first model, and the other models being tested.

Figure 7: Output of the first model (Left: Original Final Frame, right: Predicted Frame)

This figure 7 shows us the outputs of the first CNN Model. This is after the Model has completed all its runs. As one can see, the colors of the output are either not there at all, or the color is messy. It seems like this model does not perform very well, more frames or more steps could probably fix this. However, there are still rooms for improvement without having to consume time retraining it with more steps. The final frame of the output data is a pretty close representation of the actual final frame.

Table 2: Training Data of the CNN/LSTM Model

Epoch	Loss	Perceptual Distance	val_loss	Val_perceptaual distance
1	0.0176	86.3844	0.0071	29.5442
2	0.0156	81.0326	0.0026	28.5307
3	0.0149	78.4154	0.0069	25.3183
4	0.0153	78.4154	0.0086	26.3985
5	0.0142	75.4343	0.0061	26.9848
…				
45	0.0063	38.9269	0.0042	29.8360
46	0.0060	38.1099	0.0036	26.8253
47	0.0062	38.6738	0.0030	24.8659
48	0.0071	39.7716	0.0044	28.0764
49	0.0061	38.4688	0.0045	28.8115
50	0.0073	40.3392	0.0068	28.4198
…				
95	0.0068	37.5236	0.0042	27.1881
96	0.0069	37.7011	0.0036	26.9901
97	0.0061	36.4725	0.0035	28.4020
98	0.0065	37.3174	0.0087	29.2851
99	0.0065	36.7789	0.0078	26.9723
100	0.0067	37.6015	0.0020	26.7127

This model essentially uses the same metrics that and validation used in the first model, such will be the same for the last model as well. Although the training for this model was particularly longer than the other two, it still uses 100 epochs, so comparing would be fair for all models. The Data shows that the model shows promise in being able to best the first Model on terms of the validation perceptual distance. That means it should have better coloring than the first one.

Confusion Matrix

Table 3: Confusion Matrix (Accuracy of 49%)

Confusion Matrix with Accuracy of 49.00%	
44	63
39	54

The confusion matrix provided is a 2x2 matrix, which means that there are two classes that are being predicted. The rows in the matrix shows the true class, while the columns in the table indicate the predicted class. The first number in each cell of the matrix represents the number of observations that were correctly classified, while the second number represents the number of observations that were incorrectly classified. The first cell of the matrix shows that 44 observations were correctly classified as class 1, while 63 observations were incorrectly classified as class 1. The second cell of the matrix shows that 39 observations were correctly classified as class 2, while 54 observations were incorrectly classified as class 2.

The higher the value in the diagonal cells of the matrix, the better the model is performing. In this case, the model is performing relatively well, as the number of correctly classified observations is high for both classes. However, there is still room for improvement, as the number of incorrectly classified observations is also high for both classes.

True Positive (TP): This is the value or number of observations that were considered as class 1.

False Positive (FP): This is the value or number of observations that were not considered as class 1.

True Negative (TN): This is the value or number of observations that were considered as class 2.

False Negative (FN): This is the value or number of observations that were not considered as class 2.

CONCLUSION

In this study, a MATLAB-based approach was implemented to investigate dataset testing and training for enhanced human-like frame prediction using CNNs. The code provided a framework for preprocessing, training, and evaluating the network on both images and videos. While the specific results and outcomes will depend on the dataset and training parameters used, this study demonstrated the potential of deep learning techniques for frame prediction tasks. Further research and optimization are encouraged to advance the accuracy and applicability of such approaches in diverse scenes and real-world scenarios.

REFERENCES

M. Banisharif, A. Mazloumzadeh, M. Sharbaf and B. Zamani, (2022). Automatic Generation of Business Intelligence Chatbot for Organizations, 27th International Computer Conference, Computer Society of Iran (CSICC), Tehran, Iran, Islamic Republic of, pp. 1-5, doi: 10.1109/CSICC55295.2022.9780490.

Y. Liu, X. Li and Z. Xiang, (2022). The Effect of Chatbot-customer Interaction on Consumer Brand Advocacy: Exploring the Role of Chatbots. IEEE 12th International Conference on Electronics Information and Emergency Communication (ICEIEC), Beijing, China, pp. 185-190, doi: 10.1109/ICEIEC54567.2022.9835050.

Sun, Baohua, (2020) et al. "Multi-modal Sentiment Analysis using Super Characters Method on Low-power CNN Accelerator Device." arXiv preprint arXiv:2001.10179.

Abdul-Kader, S. A., Woods, J., & Thabet, T. (2019). Automatic Web-Based Question Answer Generation System for Online Feedable New-Born Chatbot. Retrieved from

https://link.springer.com/chapter/10.1007/978-3-030-01174-1_7.

Gokaran, & Ayush. (2019). Development of Chatbot Using Deep NLP and Python. Retrieved from http://122.252.232.85:8080/jspui/handle/123456789/22777.

Breuss, Martin (2022, October 12). Chatterbot: Build a chatbot with Python. Retrieved from https://realpython.com/build-a-chatbot-python-chatterbot/#project-overview

Deshpande, A., Shahane, A., Gadre, D., Deshpande, M., & Joshi, P. (2017). A Survey of Various Chatbot Implementation Techniques. Retrieved from http://www.ijcea.com/survey-various-chatbot-implementation-techniques/.

Kumar, P., Sharma, M., Rawat, S., & Choudhury, T. (2018). Designing and Developing a Chatbot Using Machine Learning. Retrieved from https://ieeexplore.ieee.org/abstract/document/8746972.

Lee, K., Jo, J., Kim, J., & Kang, Y. (2019). Can Chatbots Help Reduce the Workload of Administrative Officers? - Implementing and Deploying FAQ Chatbot Service in a University. Retrieved from https://link.springer.com/chapter/10.1007/978-3-030-23522-2_45.

Nivethan, & Sankar, S. (2019). Sentiment Analysis and Deep Learning Based Chatbot for User Feedback. Retrieved from https://link.springer.com/chapter/10.1007/978-3-030-28364-3_22.

Nuruzzaman, M., & Hussain, O. K. (2018). A Survey on Chatbot Implementation in Customer Service Industry Through Deep Neural Networks. Retrieved from https://ieeexplore.ieee.org/abstract/document/8592630.

Patel, N. P., Parikh, D. R., Patel, D. A., & Patel, R. R. (2019). AI and Web-Based Human-Like Interactive University Chatbot (UNIBOT). Retrieved from https://ieeexplore.ieee.org/abstract/document/8822176.

Rahman, A. M., Mamun, A. A., & Islam, A. (2017). Programming Challenges of Chatbot: Current and Future Prospective. Retrieved from https://ieeexplore.ieee.org/abstract/document/8288910.

Ranoliya, B. R., Raghuwanshi, N., & Singh, S. (2017). Chatbot for University Related FAQs. Retrieved from https://ieeexplore.ieee.org/abstract/document/8126057.

Santoso, H. A., Saraswati, G. W., Rohman, M. S., Winarsih, N. A. S., Sukmana, S. E., Nugraha, A., … Firdausillah, F. (2018). Dinus Intelligent Assistance (DINA) Chatbot for University Admission Services. Retrieved from https://ieeexplore.ieee.org/abstract/document/8549797.

Serban, I. V., Sankar, C., Germain, M., Zhang, S., Lin, Z., Subramanian, S., … Bengio, Y. (2017). A Deep Reinforcement Learning Chatbot. Retrieved from https://arxiv.org/abs/1709.02349

Sheikh, S. A. (2019). Artificial Intelligence Based Chatbot for Human Resource Using Deep Learning. Retrieved from https://www.researchgate.net/publication/333389243_ARTIFICIAL_INTELLIGENCE_BASED_CHATBOT_FOR_HUMAN_RESOURCE_USING_DEEP_LEARNING_A_DISSERTATION_Submitted_in_partial_fulfilment_of_the_requirements_for_the_award_of_the_degree_of_MASTER_OF_TECHNOLOGY_in_A.

Singh, R., Paste, M., Shinde, N., Patel, H., & Mishra, N. (2018). Chatbot Using TensorFlow for Small Businesses. Retrieved from https://ieeexplore.ieee.org/abstract/document/8472998.

Swanson, K., Yu, L., Fox, C., Wohlwend, J., & Lei, T. (2019). Building a Production Model for Retrieval-Based Chatbots. Retrieved from https://arxiv.org/abs/1906.03209.