Research Internship (PRe)

Specialization: Information science & Robotics & Mechanics
Academic year: 2019/2020

Deep learning for predicting ship motion from images, pitches and rolls

This is a non-confidential and publishable report

Author:
Dajing GU

Promotion:
2021

ENSTA Paris Tutor:
Jean Boisson

Internship Tutor:
Natalia Díaz Rodríguez

Internship time: 18/05/2020 - 21/08/2020

Host Organization: ENSTA Paris

Address: 828, Boulevard des Maréchaux

Non-confidential report and publishable on the Internet

Confidentiality Notice

This is a non-confidential report and publishable on the Internet. The codes associated with this report can be used freely.

Abstract

With the continuous improvement of computing power, more and more autonomous driving systems involving artificial intelligence have now been developed, which can meet people's requirements for higher efficiency and more comfort. Marine transportation plays an important role in the increasing globalization. I worked on the problem of predicting ship motion from images of the sea surface and previous pitches and rolls.

First of all, a data set is created by using 3D graphics generator Blender, which simulates the ship's movement through sea waves and offers us the information of the boat (the parameter of pitch and roll in our case). A more professional physical platform: Unreal Engine has also been tried. Data streams were structured in sequences and normalized for further processing. Then 16 different neural networks were developed based on previous work in order to better handle the time-ordered image-data sequences. Gated Recurrent Unit (GRU), Attention mechanism and Transformer Model have been introduced in the existent models of convolutional neural networks (CNN) and long short-term memory networks (LSTM). The models are tested and the best model hyper-parameters have been found by the Hyperband algorithm. All experiment results are analyzed. Also, some possible improvements are suggested.

Keywords: Deep learning, Ship motion, Time series forecasting, Blender, Unreal Engine, LSTM, GRU, Attention, Transformer

Résumé

Aujoud'hui de plus en plus de systèmes de conduite autonomes ont été développés à l'aide de l'intelligence artificielle, qui peuvent répondre aux besoins plus strictes des personnes en matière d'efficacité et de confort. Le transport maritime joue un rôle important dans la mondialisation croissante. J'ai travaillé sur le problème de la prédiction du mouvement des navires à partir d'images de la surface de la mer et des lancers et roulis précédents.

Tout d'abord, un ensemble de données est créé à l'aide du 3D logicielle Blender, qui est une simulation du mouvement du navire dans les vagues de la mer. Blender nous offre aussi des informations du bateau. Une plateforme physique plus professionnelle: Unreal Engine a aussi été utilisée. Les données ont été structurés en séquences et normalisés pour un traitement ultérieur. Ensuite, 16 réseaux de neurones différents ont été développés sur la base de travaux antérieurs. Unités récurrentes gated (GRU), mécanisme d'attention et modèle de transformateur ont été introduits dans les modèles existants de réseaux de neurones convolutifs (CNN) et de réseaux de mémoire à long terme (LSTM). Les modèles sont testés et les hyper-paramètresle des meilleurs modèles ont été trouvé par l'algorithme d'Hyperbande. Tous les résultats de l'expérience sont analysés. De plus, certaines améliorations possibles sont suggérées.

Mots-Clés: Apprentissage en profondeur, Blender, Unreal Engine, LSTM, GRU, Attention, Transformer

Non-confidential report and publishable on the Internet

Acknowledgements

I would like to thank Natalia Díaz Rodríguez, professor of the Computer Science and System Engineering Laboratory and my internship tutor at ENSTA Paris, for her care and her constant support and advice for my work. She always helped me whenever I needed help during this project.

I am also grateful to Jean Boisson, professor of the Institute of Mechanical Science and Industrial application and my ENSTA Paris Tutor, for his care on my life during the quarantine.

I also want to thank the members of the laboratory for their patience and support in overcoming numerous obstacles I have been encountered through my project in distance.

Non-confidential report and publishable on the Internet

Contents

Abstract **5**

Acknowledgements **7**

Contents **9**

1 Introduction **12**
1.1 Background . . . 12
1.2 Internship mission . . . 12
1.3 Plan . . . 13

2 Related work **14**
2.1 Data-driven analysis . . . 14
2.2 Related work on Ocean engineering . . . 14
2.3 Related work on ship motion prediction problems . . . 15

3 Data generation **16**
3.1 Ship motions . . . 16
3.2 Data generation . . . 17
3.2.1 Blender . . . 17
3.2.2 Unreal Engine . . . 18
3.3 Data pre-processing . . . 19
3.3.1 Vectorization . . . 19
3.3.2 Value normalization . . . 19
3.3.3 Missing value . . . 19

4 Proposed models **20**
4.1 Previous work on CNN and LSTM . . . 20
4.1.1 CNN - Convolutional neural network . . . 20
4.1.2 LSTM - Long short term memory networks . . . 21
4.1.3 Existent models based on LSTM and CNN . . . 22
4.2 Principle of new models . . . 23
4.2.1 GRU - Gated recurrent unit networks . . . 23
4.2.2 Attention Mechanism . . . 24
4.2.3 Transformer . . . 25
4.3 Models Nomenclature . . . 25
4.3.1 Models Nomenclature . . . 25
4.4 Comparison between models with sequence length of size 1 and models with sequence length of size N . . . 26
4.4.1 Difference in the part of encoder . . . 27
4.4.2 Difference in the part of decoder . . . 28
4.5 Models with sequence length of size 1 . . . 29
4.5.1 GRU encoder decoder PR model . . . 29
4.5.2 LSTM encoder attention decoder PR model . . . 29

4.5.3 GRU encoder attention decoder PR model 30
4.5.4 CNN GRU encoder decoder images PR model 30
4.5.5 CNN LSTM encoder attention decoder images PR model 31
4.5.6 CNN GRU encoder decoder images PR model 31
4.5.7 CNN LSTM encoder GRU attention decoder images PR model . . 32
4.6 Models with sequence length of size N 32
4.6.1 LSTM encoder decoder PR* 33
4.6.2 GRU encoder decoder PR* 34
4.6.3 LSTM encoder attention decoder PR* 34
4.6.4 GRU encoder attention decoder PR* 34
4.6.5 LSTM encoder GRU attention decoder PR* 34
4.6.6 CNN LSTM encoder decoder images PR* 35
4.6.7 CNN LSTM encoder attention decoder images PR* 35
4.6.8 CNN LSTM encoder GRU attention decoder images PR* 36
4.7 TransformerModel PR model 36

5 Model Training **38**
5.1 Parameters 38
5.2 Data loading 38
5.3 Loss function and Optimizer 39
5.4 Underfitting and Overfitting problem 40
5.5 Hyperparameters adjustment 40

6 Model Testing **41**
6.1 Models with sequence length of size 1 41
6.1.1 Models without a CNN part 41
6.1.2 Models with a CNN part 42
6.2 Models with sequence length of size N 44
6.3 Model without direct use of RNN model 45
6.4 Models comparison 45
6.4.1 Comparison between the models with different sequence length . . 46
6.4.2 Comparison between the models the best performance 46
6.5 Relation between the size of Past sliding window and Predicted sliding window 49
6.6 Results 49

7 Conclusion **51**
7.1 Improvements 51

Bibliography **53**

List of figures **55**

List of tables **57**

A Architectures of models **59**
A.1 Architectures of models with sequence length of size 1 59
A.2 Architectures of models with sequence length of size N 62

1 Introduction

1.1 Background

With the continuous improvement of computing power, more and more autonomous driving systems involving artificial intelligence have now been developed. This kind of self-driving system can meet people's requirements for higher efficiency and more comfort. In addition to the transportation on roads like trains and trucks, marine transportation also plays an important role in the increasing globalization and the rapid growth in international trade [1]. The auto-navigation system has a wide prospect for marine transportation, especially in aircraft landing, cargo transfer, off-loading of small boats, and ship "mating" between a big transport ship and some small ships.

Different from the working environments of cars, where lots of attention is paid to the cars and pedestrians, many other parameters like ship motion should be taken into consideration in the design of an auto-pilot system for ships. Ship motion can improve the efficiency of several naval operations. A five to ten seconds of ship motion prediction is able to ensure the accuracy of the coordination to avoid damage to boats or aircraft, and to ensure the safety of operations by reducing all risks to a minimum [2]. In order to have better control of the ships in real time, reconstruction of ocean waves, navigational state prediction and other technologies should be used.

Therefore, in order to design a better autopilot system for vessels, this internship has focused on the problem of ship motion prediction, which is considered as a time series forecasting problem, in using artificial intelligence methods like CNN (Convolutional Neural Network), LSTM (Long Short-Term Memory neural network), GRU (Gate Recurrent Unit) and Attention mechanism.

1.2 Internship mission

As is mentioned in the part of background, the goal of the internship is to develop deep learning models (many-time-step models, GRU, Attention mechanism and Transformer model) based on existent models (CNN and LSTM) to predict vessel movements from sea surface images and ship motion states, in hoping that 25 seconds' ship motion would be able to be predicted in advance as accurately as possible.

Reconstruction of ocean waves are useful in the aspect of real-time ship motion forecasting because of the correlation between wave behaviour and ship motion. For lack of real data, videos of sea surface and the correspondent state of the ship are created by the wave simulator and then used as input of the deep learning models. After training and analyzing the training result, we would enable the neural network model to predict the state of the ship from sea surface photos and past ship motion data.

The mission in this internship is also shown in **Figure 1.1**.

Non-confidential report and publishable on the Internet

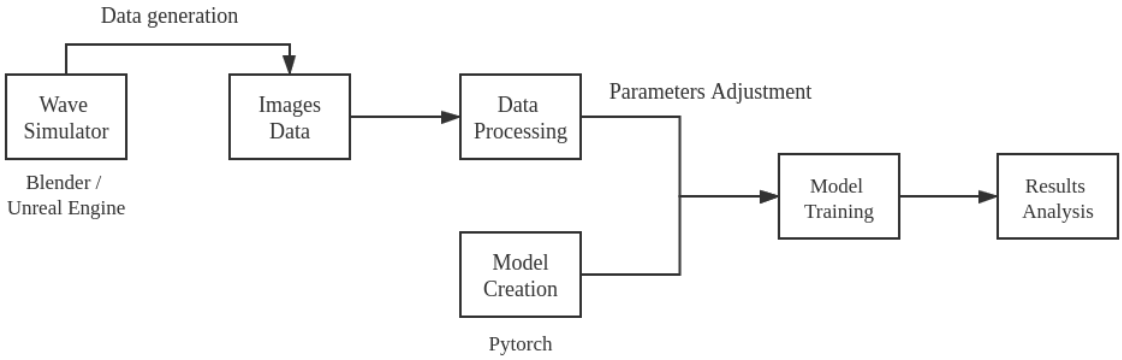


Figure 1.1: Internship mission

1.3 Plan

The report is organized as follows: Chapter 2 gives an presentation of the related work of data-driven method in the field of ocean engineering and in the field of time series forecasting problem. Chapter 3 gives an understanding of the data we generated to work with. The models involved in this internship and the overview of deep learning will be introduced in Chapter 4, and training aspects in Chapter 5. Test results and analysis of the results can be found in Chapter 6. The conclusion will be drawn and some ideas for improvement will be given in Chapter 7.

Non-confidential report and publishable on the Internet

2 Related work

In this chapter, the related work of the internship is divided into three parts: an overview of the conception of data-driven analysis, precedent work on ocean engineering. and finally related work in ship motion prediction problems.

2.1 Data-driven analysis

Machine learning (ML) is a discipline focusing on data analysis. In Machine learning algorithms, a mathematical model is built based on sample data, known as "training data", and the algorithms can be improved automatically in order to make predictions or decisions without being explicitly programmed to do so.

Among the ML algorithms, Deep learning (out of neural network) is one of the most important breakthroughs in the field of artificial intelligence in the past decade, which has achieved great success in speech recognition, natural language processing, computer vision, image and video analysis and many other fields.

Thanks to the increasing computing power and the big quantity of data accumulated, data-driven analysis method has given scientists a golden chance to study again many traditional problems. In spite of the potential shortcomings of ML, like the problem of over-fitting, many new conclusions are drawn and our life are being improved.

In 2019, Cipollini et al. [3] used DNN (deep neural network) to deal with the problem of assessing the health condition of induction motor bearings and proved the more powerful effectiveness of deep neural network than classic techniques like the PCA. Later in 2019, Oneto et al. [4] combined an experience-driven method and a data-driven method to deal with the problem of understanding and predicting the train movements in large-scale railway net works, which outperforms both state-of-the-art experience-based and data-driven systems.

2.2 Related work on Ocean engineering

As is mention above, data driven method is an effective and efficient way for scientists to find new rules and draw conclusions. This kind of state of the art method is also used in ocean engineering.

In 2008, Zamani et al. [5] presented a number of data-driven models for wind–wave process at the Caspian Sea. Models based on artificial neural network (ANN) and instance-based learning (IBL) showed better function on forecasting significant wave heights for several hours ahead using buoy measurements.

In 2018, Oehmcke et al. [6] designed a deep learning architecture for a marine virtual sensor application that is able to utilize input quality information. They proposed a novel input quality based dropout layer to take advantage of Information about the quality of the input sensors, which overcomes the baseline of bLSTM layers with a time dimensionality reduction method called exponential piecewise aggregate approximation (exPAA) .

Non-confidential report and publishable on the Internet

2.3 Related work on ship motion prediction problems

The data-driven analysis method is widely used in ocean engineering, however, seldom are they used to predict waves or the motion of a ship. Given the importance of ship motion and wave behaviour, in the internship, deep learning models will be designed to predict ship motion according to the wave behaviour and past ship motion data.

In 2004, George Zhao et al. [2] compared the effect of minor component analysis (MCA) and neural networks (NN), auto-regressive methods (AR), and Wiener prediction in the development of a high performance ship-motion prediction algorithm. Simulation results showed that MCA can predict ship motion a long time ahead with consistent accuracy.

The main idea of this internship is a continuation of work presented in [7] and [8]. The two authors have offered some solutions using artificial intelligence methods like Convolutional Neural Network and Long Short Term Memory neural network, where the model of auto-encoder has been utilized. However, the too long inputs may cause sharp decline of the effect of the models. Under this circumstance, the Attention mechanism is considered to be used.

Attention mechanism is firstly used in computer vision problem [9] and then widely used in translation, speech recognition, image caption (Image Caption) and many other fields. In 2015, Xu et al. [10] introduced attention in Image Caption. In order to better capture the long-term temporal dependencies appropriately and to select the relevant driving series to make predictions, in 2017, Qin et al. [11] proposed a dual-stage attention-based recurrent neural network (DA-RNN) which can outperform state-of-the-art methods for time series prediction. In 2018, Tharindu et al. [12] proposed a novel method to predict the future motion of a pedestrian, in combining hard attention and soft attention. And in the same yaer, Tharindu et al. [13] has introduced a novel pedestrian detection platform based on Generative Adversarial Networks (GANs) to resolve the over reliance on the use of appearance models in current multi-person localisation and tracking systems.

In spite of the effect of attention mechanism in time series forecasting problems,the problem of long term dependencies of RNN (recurrent neural networks) can not be avoided. In 2017, Vaswani et al. [14] has introduced a new model named Transformer, which has very good effect in many aspects without a direct utilization of the RNN model.

In consideration of the effectiveness of the Attention mechanism and the transformer Model, this internship will focus on the creation of them on the existent models.

Non-confidential report and publishable on the Internet

3 Data generation

Prediction of ship motion helps a lot for us to achieve the goal of ship auto-navigation. In this chapter, types of ship motion and different input data generation will be presented. This is a typical sequence-to-sequence problem [15], of which the input will be sets of wave images as well as the labels of ship motions, the output is also a sequence.

3.1 Ship motions

Firstly different types of ship motions are introduced. Ship motions are defined by the six degrees of freedom that a ship, boat or any other craft can experience [2]. According to the difference, ship movements are divided into two groups: [1]

- Rotational motions

– Pitch: The up/down rotation of a vessel about its transverse Y axis.

– Roll: The tilting rotation of a vessel about its longitudinal X (front-back or bow-stern) axis.

– Yaw: The turning rotation of a vessel about its vertical/Z axis.

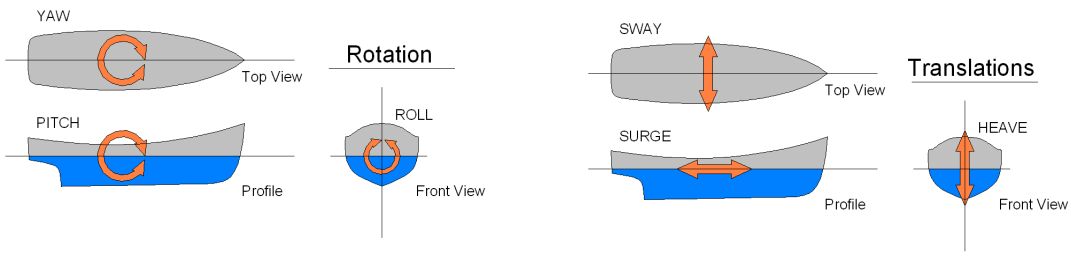


Figure 3.1: Illustration of rotational and translational motion from wikipedia

- Translational motions

– Heave: The linear vertical (up/down) motion; excessive downward heave can swamp a ship

– Sway: The linear transverse motion.

[1] https://en.wikipedia.org/wiki/Ship_motions

– Surge: The linear longitudinal (front/back or bow/stern) motion imparted by maritime conditions.

In the internship, it's the rotational motion that should be attached importance to since the translational motion: heave, sway and surge are more difficult to measure and relatively less important. In the three parameters of rotational motion, it's the pitch and roll that should be paid attention to because yaw is mainly controlled by captain or GPS module to get into the destination, which is fixed in our case to follow a certain like a circle.

3.2 Data generation

In order to get better training results, it is very important to have a lot of training data. It is not difficult to collect data theoretically with a stabilized camera installed on the ship to record the sea conditions in front of the ship along with the navigation state information of the ship collected in real time. However, for lack of equipment and conditions (quarantine of COVID-2019), 3D software are used in this internship to generate the input data and images.

3.2.1 Blender

Firstly I have a try with Blender. Blender is a professional, free and open-source 3D computer graphics software used for creating animated films, visual effects, art and video games, etc. Blender's features include 3D modeling, texturing, animating, match moving, camera tracking, rendering, motion graphics and so on.

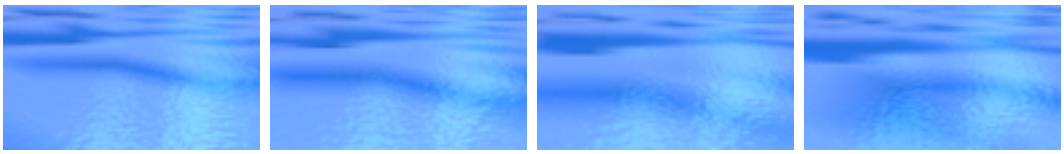


Figure 3.2: Illustration of images generated by Blender

Thanks to the work presented [2] [3], the scripts of Python API can be used to generate the necessary data. The code in the script has been updated by me to adapt to the newest version of Blender 2.83. All needed is to specify the necessary parameters of the ocean, the camera position and an image expansion. In this case:

- Camera is fixed on the simulated ship with next parameters:

height	rotation x	rotation y	rotation z
5 m	76 rad	90 rad	75 rad

Table 3.1: Camera parameters in data generation

[2] https://github.com/Nazotron1923/ship-ocean_simulation_BLENDER
[3] https://www.youtube.com/watch?v=sfi7HW8qHAo

Non-confidential report and publishable on the Internet

- Image resolution

height	width
54 pixels	96 pixels

Table 3.2: Image resolution

Other ocean parameters as choppiness, wave scale, wind velocity, and wave alignment were chosen randomly, in a realistic looking manner, to add the complexity of the sea states and to better simulate the sea wave models. The data generated by Blender are then transformed to the form needed for training. Finally, 670 episodes of data are created, with 400 pictures in each episode, which means the data of 37.2h are created.

3.2.2 Unreal Engine

Even though the simulation of waves offered by blender helps a lot, many physical parameters can not be well simulated in blender, which may influence our training results. In order to avoid this kind of loss, I have a try with a more powerful and more realistic platform: Unreal engine .

Unreal Engine is the world's most open and advanced real-time 3D creation platform for photo-real visuals and immersive experiences. The physique parameters can be saved during the simulation.

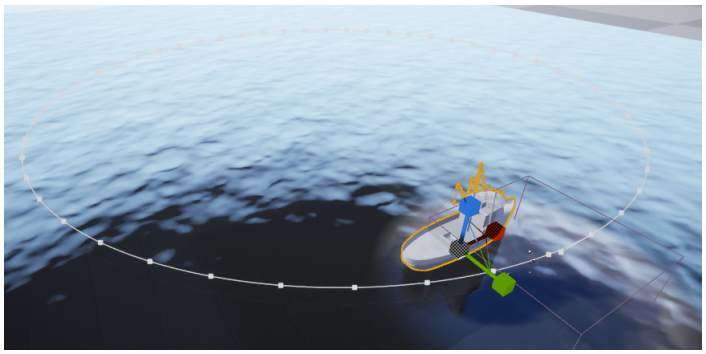


Figure 3.3: Illustration of ocean models generated by Unreal Engine

Different from the creation of ocean in Blender[4], creation of ocean in UE4 (Unreal Engine) is much more complex. The free ocean materials can't give us a perfect form of waves and the buoyancy effect. Even though a simple ocean model with some physical conditions like color, the reflection, the textures (presented in the picture) have been created, there lacks tutorials for the wave forms and the buoyancy effect (the existed tutorials always based on paid plugin), which means that the pitch and roll angel can be obtained, and that boat can only move in a horizontal plane. So I return to Blender for help.

[4] https://www.youtube.com/watch?v=sfi7HW8qHAo&t=227s

Non-confidential report and publishable on the Internet

3.3 Data pre-processing

In order to make the original input data more suitable for neural network training, data pre-processing methods are needed [16] , among which the most popular are vectorization, standardization, normalization and processing of missing values.

3.3.1 Vectorization

No matter what kind of data is processed (sound, image, or text), the first step is to convert it into a tensor of floating-point data type. All inputs should be converted into the same size. This operation is not needed in the internship since all the input images are created automatically and are of the same size.

3.3.2 Value normalization

Convert the mean of the data to 0 and its standard deviation to 1. By using the same features and the same range of data, the input of the neural network is made stable. There will be fewer large gradient updates which makes the network learn easier.

For the images generated by Blender, each pixel is converted into values between -1 and 1, whose formula is below.

$$P_{normalized} = \frac{2 \times P_{original}}{255} - 1$$

As for pitch and roll values for some models, since the angels are always between -90 and $+90$, the following formula is used to normalize the data in [-1, 1]:

$$angle_{normalized} = \frac{angle_{original} - (-90)}{90 - (-90)} \times 2 - 1$$

3.3.3 Missing value

Sometimes some missing values should be set 0 to ensure the effect of learning and the updating of weights. Which is also unnecessary in this internship.

Non-confidential report and publishable on the Internet

4 Proposed models

The internship mission is a time series forecasting problem, with the images of wave and precedent ship motion data as input, for which CNN - Convolutional neural networks [17] and LSTM - Long short-term memory networks [18] will be the most appropriate models to be used and some models have already been created in previous work (see **4.1**).

Baesd on the extent models, 16 new models are created. GRU - Gated recurrent unit networks [18], Attention mechanism [9] (see **4.2.1**) are used. The structure of the extent models are changed from sequence length of size 1 to sequence length of size N (see **4.5**) in order to make full use of Attention Mechanism (see **4.2.2**). Transformer Model [14] (see **4.2.3**) are also considered to be used. An overview of all the models above will be given, and the proposed model created in this internship will also be analyzed.

4.1 Previous work on CNN and LSTM

The input of the models will be the images of sea waves and the values of pitches and rolls in a past period of time, so CNN and LSTM will be the most appropriate models to be used and some models have already been created in previous work [8].

4.1.1 CNN - Convolutional neural network

In the internship, CNN will be used since the input data are images of waves. Convolutional neural network is a neural network containing convolutional layers, which has 2 major characteristics:

1) Can effectively reduce the size of large data pictures to small data

2) Can effectively retain the characteristics of the picture, in line with the principles of picture processing

The layers of CNN can be divided into 3 groups: Convolutional layer, pooling layer and a fully connected layer.

The convolution layer is used for feature extraction. The entire picture is scanned with a convolution kernel. This process can be understood as we use a filter (convolution kernel) to filter each small area of the image to obtain the characteristics of these small areas value. The pooling layer is simply down-sampling, which can greatly reduce the dimension of the data.

The pooling layer can reduce the data dimension more effectively than the convolution layer. Not only can this greatly reduce the amount of calculation, but it can also effectively avoid over-fitting.

As for the output layer, the data processed by the convolution layer and the pooling layer are input to the fully connected layer to obtain the final desired result. Only after the dimensional reduction of the convolution layer and the pooling layer can the fully connected layer operate normally, otherwise the amount of data is too large and the calculation cost is too high.

Non-confidential report and publishable on the Internet

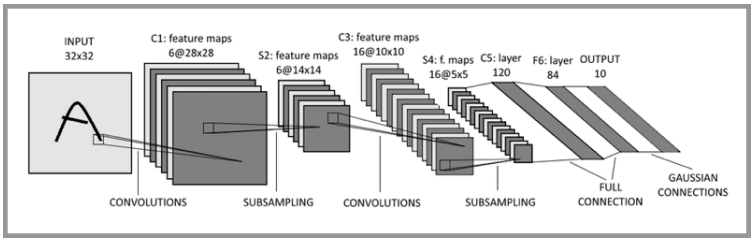


Figure 4.1: Illustration of LeNet

As a representative algorithm for deep learning, convolutional neural networks have the ability to represent learning, that is, they can extract high-order features from input information. In a successful convolutional neural network, the feature map passed to the fully connected layer will contain the same features as the learning target, such as the complete image of each category in the image classification [19].

4.1.2 LSTM - Long short term memory networks

Long Short-Term Memory networks, usually just called "LSTMs" – are a special kind of RNN, capable of learning long-term dependencies[5].

One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. RNNs can meet our requirements when only the recent information is needed. For example, predicting the next word based on the previous ones. However, when more context is needed, RNNs become unable to connect the information.

The problem was explored in 1991 and LSTM is designed. LSTM can avoid the long-term dependency problem and remembering information for long periods of time.

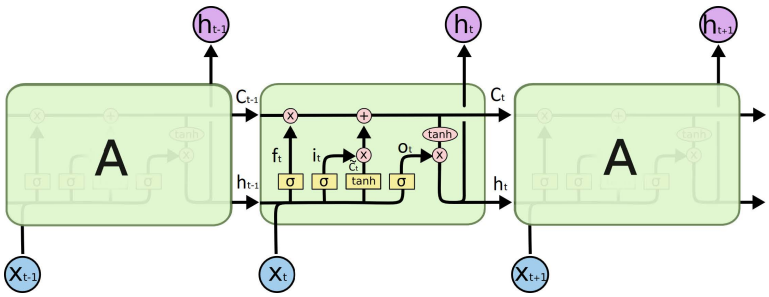


Figure 4.2: The repeating module in an LSTM contains four interacting layers: 1) sigmoid layer (f_t); 2) sigmoid layer (i_t); 3) tanh layer ($\tilde{C}_t$); 4) sigmoid layer (o_t).

LSTM has the form of a chain of repeating modules of neural networks. The repeating module has four interacting layers in a very special way. Cell state is the most important

[5] https://colah.github.io/posts/2015-08-Understanding-LSTMs/

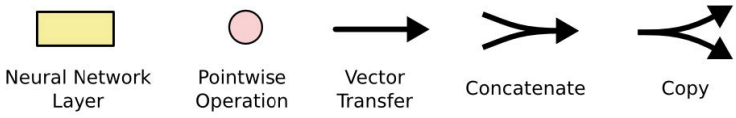


Figure 4.3: LSTM notions

part of the LSTM architecture, here we can store information extracted from input, add new or delete old. All this is allowed by a special structure called gates.

The forget gate layer of type sigmoid is introduced to forget the information that is not important.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

Then to decide what new information to be stored in the cell state. This procedure is divided into two parts. Firstly, the input gate layer of type sigmoid layer called the "r" decides which values we'll update. Secondly, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. These two are combined to create an update to the state.

$$\begin{aligned} i_t &= \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right) \\ \tilde{C}_t &= \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right) \end{aligned}$$

The information is updated as follows: certain information $f_t \cdot C_{t-1}$ are forgotten and a new memory $i_t \cdot \tilde{C}_t$ is added.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Finally, an output based on our cell state will be given, but in a filtered version.

$$\begin{aligned} o_t &= \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right) \\ h_t &= o_t * \tanh\left(C_t\right) \end{aligned}$$

4.1.3 Existent models based on LSTM and CNN

In previous work, some models are already created based on CNN and LSTM [8]. Here the architecture of **LSTM encoder decoder PR** and **CNN LSTM encoder decoder images PR** are presented, precise illustration of the architecture can be seen in **A.1**.

Since the internship mission is considered as a time series forecasting problem, in which a sequence of past frames is used to predict a sequence of the following future frames, a sequence to sequence prediction approaches is chosen in this internship. In previous work [6], an encoder-decoder architecture from Pytorch tutorial was implemented, whose architecture can be seen in the table below. Here the 300 is the size of the hidden units of both the encoder and the decoder. The output of the encoder will be directly the input of the decoder. The initial hidden states of both the encoder and the decoder will be zero tensor.

[6] https://github.com/Nazotron1923/PRE-summer-2019-/tree/master/Pre

Non-confidential report and publishable on the Internet

Input	Sequence of N pitches and rolls $[1 \times \mathbf{N} \cdot 2]$					
Name	*Layer*	*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	LSTM	1	Ture	1×300	$1 \times \mathrm{N} \cdot 2$	1×300
Decoder	Relu + LSTM	1	True	1×300	1×300	1×300
FC 1	FC + ReLu				1×300	1×150
Output	FC + Tanh				1×150	$1 \times \mathrm{M} \cdot 2$

Table 4.1: LSTM encoder decoder PR model architecture in previous work

Then in a CNN part is added the model. This time the input is not only the values of pitch and roll but also the images of sea waves. The architecture is shown in the table below. An illustration can be found in Nazar's work[8] and in **A.1**.

Input	Sequence of N frames $[54 \times 96 \times 3]$						
N CNN parts							
Name	*Layer*	*Number filters*	*Filter size*	*Stride*	*Padding*	*Inpout size*	*Output size*
Conv 1	Convolution ReLu	8 -	3×3 -	1 -	2 -	$54 \times 96 \times 3$ -	$54 \times 96 \times 8$ -
Pooling 1	Max pooling	1	3×3	2	1	$54 \times 96 \times 8$	$27 \times 48 \times 8$
Conv 2	Convolution ReLu	16 -	3×3 -	1 -	1 -	$27 \times 48 \times 8$ -	$27 \times 48 \times 16$ -
Pooling 2	Max Pooling	1	3×3	2	1	$27 \times 48 \times 16$	$14 \times 24 \times 16$
Conv 3	Convolution ReLu	32 -	3×3 -	1 -	1 -	$14 \times 24 \times 16$ -	$14 \times 24 \times 32$ -
Pooling 4	Max Pooling	1	3×3	2	1	$14 \times 24 \times 32$	$7 \times 24 \times 32$
FC Mu	FC + ReLu					1×2688	1×1024
FC Sigma	FC + Tanh					1×2688	1×1024
RNN Part: Adding sequence of N pitches and rolls $[N \times 2]$							
Name	*Layer*		*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	LSTM		1	Ture	1×300	$1 \times \mathrm{N} \cdot 1024 + 2$	1×300
Decoder	Relu + LSTM		1	True	1×300	1×300	1×300
FC 1	FC + ReLu					1×300	1×150
FC 2	FC + Tanh					1×150	1×2
Output							$1 \times M \cdot 2$

Table 4.2: CNN GRU encoder decoder PR model architecture

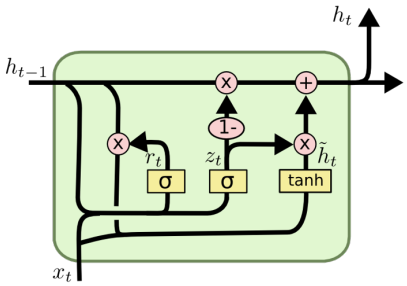
4.2 Principle of new models

In this section, GRU - Gated recurrent unit networks, Attention Mechanism and Transformer Model will be introduced.

4.2.1 GRU - Gated recurrent unit networks

Gated recurrent units (GRUs) are a gating mechanism in recurrent neural networks[7], which is like a long short-term memory (LSTM) but without an output gate. The performance of GRU is similar to that of LSTM on certain tasks of polyphonic music modeling, speech signal modeling and natural language processing [20]. What's more, GRUs have been shown to exhibit even better performance on certain smaller and less frequent datasets [21], which is just our case, so in this internship, the GRU is considered to be used as a complentary model of LSTM.

[7] https://en.wikipedia.org/wiki/Gated_recurrent_unit



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 4.4: Architecture of GRU module

4.2.2 Attention Mechanism

In the deep learning model of the internship, the model of auto-encoder has been utilized. A set of 10 images has been used as input data, however, in order to predict the ship motions in the future for a longer time, a bigger image set should be used, which may cause same problems:

1. When the input sentence length is too long, especially when it is longer than the original sentence length in the training set, the performance of the model sharp decline.

2. This is unreasonable to assign the same weight to each word in the sentence, and the performance of the model is often degraded.

Under this circumstance, the Attention mechanism is introduced.

Attention is a mechanism based on the effect of the encoder + decoder model of RNN (LSTM or GRU), which is generally called the attention mechanism. It's widely used in translation, speech recognition, image caption (Image Caption) and many other fields. The reason why it is so popular is because Attention gives the model the ability to distinguish, for example, in machine translation, each word in the sentence is given different weights, so that the learning of the neural network model becomes more flexible (soft), and the attention itself can be used as an alignment relationship to explain the input/ output sentences. The alignment relationship explains what the model has learned. It opens a black box for deep learning and provides a window.

Attention Mechanism is very similar to the human observation mechanism for external things. When humans observe external things, they generally do not look at things as a whole, and tend to selectively obtain some important parts of the observed things according to their needs. For example, when we see a person, we often focus on the person's face first, and then combine the information from different regions to form an overall impression of the observed thing. Therefore, the Attention Mechanism can help the model to assign different weights to each part of the input X, extract more critical and important information, make the model make more accurate judgments, and will not bring greater calculation and storage of the model, which is why the Attention Mechanism is so widely used.

In this internship, an Attention layer is add to the existent models to solve the problems of long term dependencies

Non-confidential report and publishable on the Internet

4.2.3 Transformer

The Transformer is a deep learning model introduced designed to handle sequential data [14], such as natural language, for tasks such as translation and text summarization[8] . However, unlike RNNs, Transformers do not require that the sequential data be processed in order and it allows for much more parallelization than RNNs and therefore the training time can be reduced.

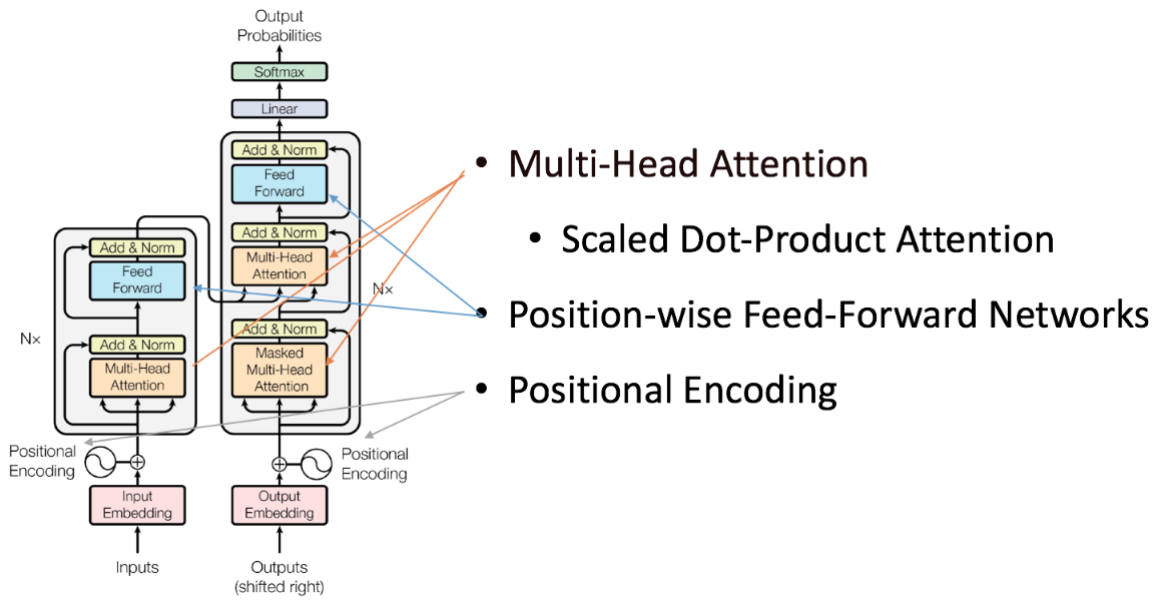


Figure 4.5: Illustration of the architecture of a transformer model

Since their introduction, Transformers have become the model of choice for tackling many problems in NLP, replacing older recurrent neural network models such as the long short-term memory (LSTM). In this internship, a transformer model has also been created in order to improve the accuracy of prediction.

4.3 Models Nomenclature

4.3.1 Models Nomenclature

In this section, the models that have been created will be described, of which the architectures and features will be analyzed. Before the description of the models, some notations helping identify a particular model by its name are introduced (see **Table 4.3**).

[8] https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)

Notation	Explanation
N	Number of past frames used as input (in the stream history sliding window)
M	Number of future predicted frames by the model
CNN	A model containing a CNN architecture
LSTM	A model containing a LSTM architecture
GRU	A model containing a GRU architecture
Transformer	A model containing a Transformer architecture
images	A model where images are input and used for prediction
PR	A model where the original pitch and roll are used as input for its prediction M frames in the future
encoder	An architecture among LSTM, GRU and Transformer is used as part of the model to encode an input vector
decoder	An architecture among LSTM, GRU and Transformer is used as part of the model to decode a latent vector

Table 4.3: Models nomenclature for each module component

The models created are separated into three parts: models with sequence length of size 1, models with sequence length of size N and models without the direct use of RNN.

Model Type	Model Name
Models with sequence length of size 1	GRU encoder decoder PR
	LSTM encoder attention decoder PR
	GRU encoder attention decoder PR
	CNN GRU encoder decoder images PR
	CNN LSTM encoder attention decoder images PR
	CNN GRU encoder attention decoder images PR
	CNN LSTM encoder GRU attention decoder images PR
Models with sequence length of size N	LSTM encoder decoder PR*
	GRU encoder decoder PR*
	LSTM encoder attention decoder PR*
	LSTM encoder GRU attention decoder PR*
	GRU encoder attention decoder PR*
	CNN GRU encoder decoder PR*
	CNN LSTM encoder attention decoder PR*
	CNN LSTM encoder GRU attention decoder images PR*
Model without RNN	TransformerModel_PR

Table 4.4: Proposed Models

4.4 Comparison between models with sequence length of size 1 and models with sequence length of size N

The difference between the models with sequence length of size 1 and models with sequence length of size N exists both in the encoder and the decoder. Here N is the number of frames we used as the input of the models. We take **LSTM encoder decoder PR** as an example in the comparison below.

Non-confidential report and publishable on the Internet

4.4.1 Difference in the part of encoder

In the models of size 1, N frames are concatenated in one sequence as the encoder input, with 2 features (pitch and roll) for each frame. So the encoder input is one sequence with $N \cdot 2$ words, whose shape is $[batchsize,\ \mathbf{1},\ N \cdot 2]$. And the size of encoder output is $[batchsize,\ \mathbf{1},\ *]$, where $*$ is the latent hidden size (embedding size) we set for the encoder.

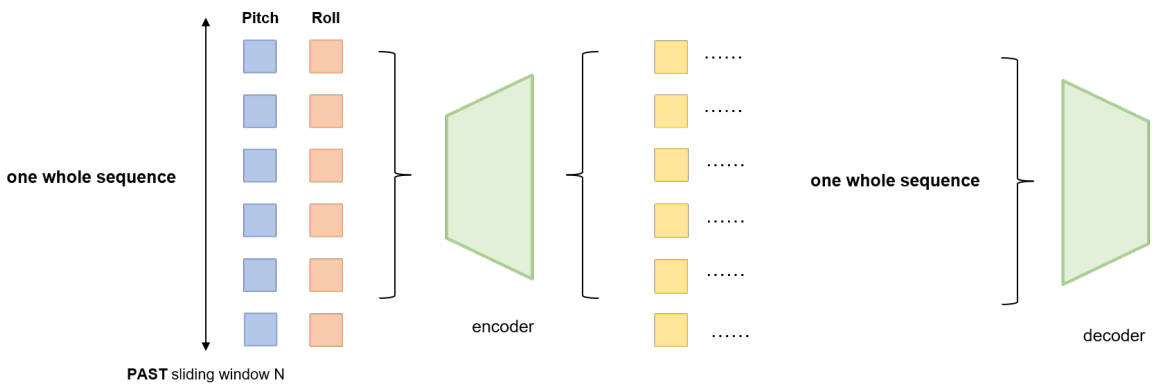


Figure 4.6: Illustration of the encoder of models with sequence length of size 1

And for models of size N, N frames are used respectively as input in N sequences. The encoder input is N sequences, with 2 words (pitch and roll) in each sequence. So the encoder input is of shape $[batchsize,\ \mathbf{N},\ 2]$, and the encoder output is of size $[batchsize,\ \mathbf{N},\ *]$, where $*$ is the latent hidden size (embedding size) we set for the encoder.

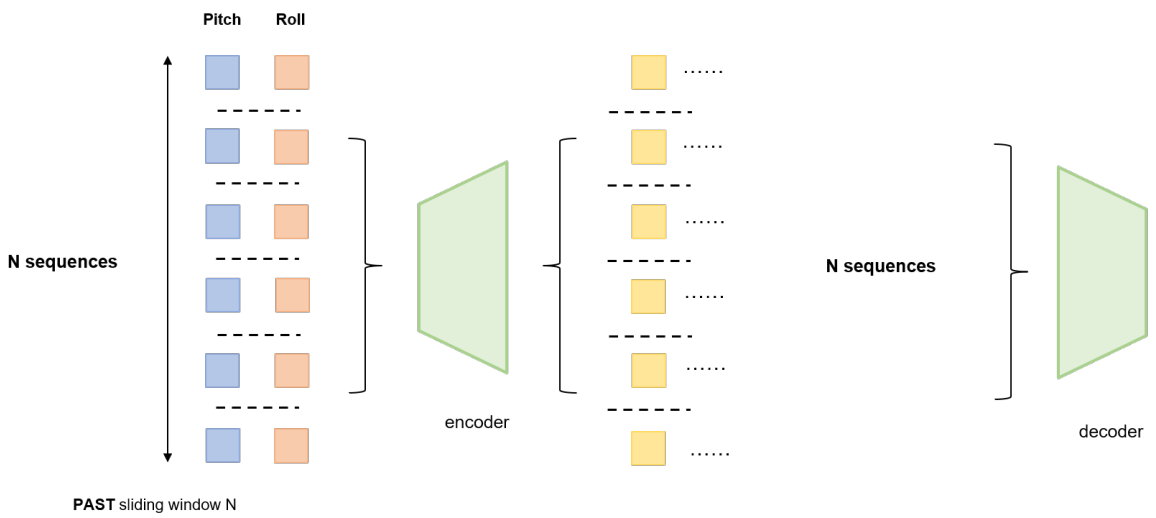


Figure 4.7: Illustration of the encoder of models with sequence length of size N

The output of a RNN model is the ensemble of the hidden states of all time steps. So in models of size 1, there is only one hidden state can be used, and for new models,

there are N hidden states can be used, which caters to the requirements of Attention Mechanism.

4.4.2 Difference in the part of decoder

As for the decoder, there are also some differences.

In models of size 1, M frames are predicted in only one step, which means the hidden states are used only once when M frames are predicted. A step in the models of size 1 is replaced with many steps in the models of size 1.

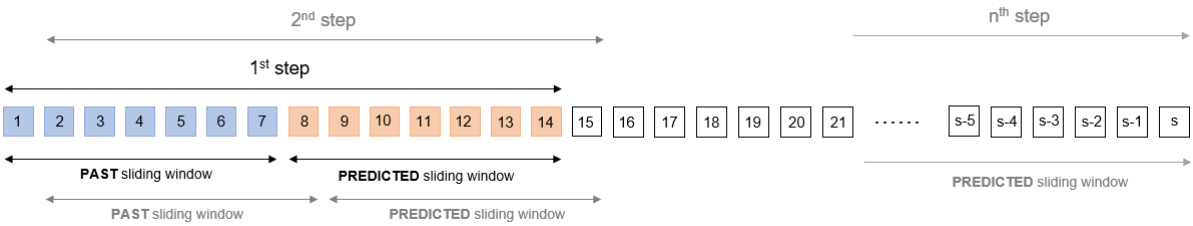


Figure 4.8: Illustration of the decoder of the models with sequence length of size 1

For models of size N, instead of predicting the result at once, the M frames are predicted frame by frame, the previous hidden states are used and updated when each frame in the future is predicted, which can satisfy the requirements of Attention Mechanism.

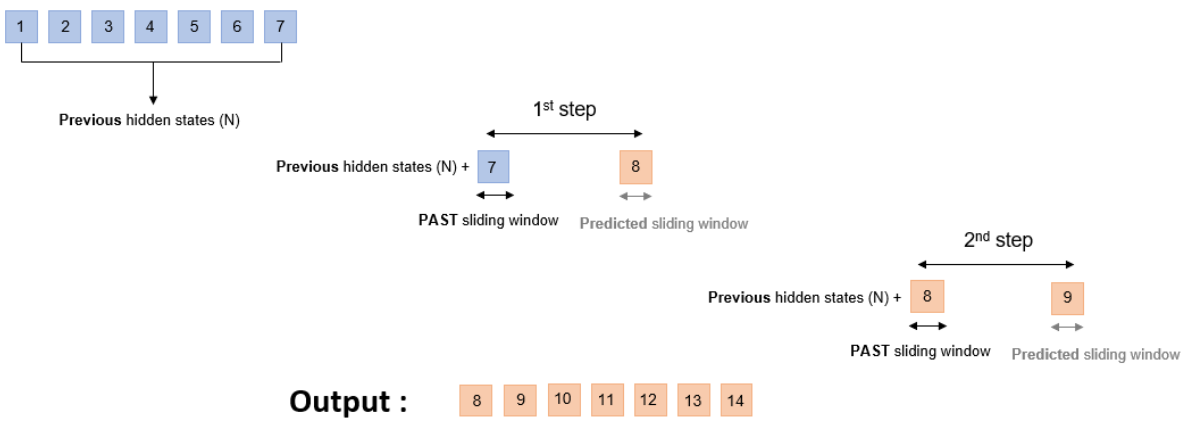


Figure 4.9: Illustration of the decoder of the models with sequence length of size N

The last encoder input will be the first decoder input of the models to predict one frame in the future. Then the newest decoder output will be the new decoder input of next step. After a cycle of M times, the M decoder output will be concatenated as the final result of prediction. The Attention Mechanism is able to connect the present frame with the hidden states of the previous hidden states. So an attention layer added in the decoder can make better use of the hidden states as well as update them in each step of the cycle.

4.5 Models with sequence length of size 1

In this section, models with sequence length of size 1 will be presented, which are based on the architecture of models created in previous work of Nazar [8]. Here size 1 means the encoder output of size [*batchsize*, **1**, $*$], where $*$ can be replaced by any number. The encoder output is the ensemble of the hidden states of all time steps. In the model with sequence length of size 1, there is a only one hidden state of only one time step. For detailed information, see **4.4** .

4.5.1 GRU encoder decoder PR model

GRU is considered to have better performance than LSTM for some certain small data set, so in this internship, some models based on GRU are created, whose architecture are very similar to those of LSTMS. In the table below, the architecture of GRU encoder decode PR is presented. The output of the encoder will be used directly as the input of the decoder, precise illustration of the architecture can be seen in **A.1**.

Input	Sequence of N pitches and rolls [$1 \times \mathbf{N} \cdot 2$]					
Name	*Layer*	*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	GRU	1	Ture	1×300	$1 \times \mathrm{N} \cdot 2$	1×300
Decoder	Relu + GRU	1	True	1×300	1×300	1×300
FC 1	FC + ReLu				1×300	1×150
Output	FC + Tanh				1×150	$1 \times \mathrm{M} \cdot 2$

Table 4.5: GRU encoder decoder PR model architecture

4.5.2 LSTM encoder attention decoder PR model

Attention mechanism can solve the problem of long-term dependency, so in this internship, an Attention layer is added to the existent model, whose architecture can be found in **Table 4.4**, precise illustration of the architecture can be seen in **A.1**.

Input	Sequence of N pitches and rolls [$1 \times \mathbf{N} \cdot 2$]					
Name	*Layer*	*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	LSTM	1	Ture	1×300	$1 \times \mathrm{N} \cdot 2$	1×300
Attention	Tanh + FC				1×600	1×300
v	Soft + FC				1×300	1×1
Decoder	Relu + LSTM	1	True	1×600	1×600	1×300
FC 1	FC + ReLu				1×600	1×150
Output	FC + Tanh				1×150	$1 \times \mathrm{M} \cdot 2$

Table 4.6: LSTM encoder attention decoder PR model architecture

The implementation of the Attention mechanism is accomplished by the linear layer. By concatenating the encoder output (which is the ensemble of hidden states of all time steps) and the new decoder input, the model can pay more attention to the previous hidden states. Even though the hidden layer is of size 1 in this model, which means in the type of [*batchsize*, **1**, $*$], an attention layer is still added to have a look at the effect.

Non-confidential report and publishable on the Internet

The architecture of the attention layer can be seen in **Table 4.5**:

Encoder			
Attention	Input of Attention Layer		
	Name	s: last hidden state	encoder output
	Size	[1, batch size, 300]	[batch size, 1, 300]
	Intermediate Parameters		
	Name	s s repeat	energy
	Formula	s.repeat(seq len,1,1)	tanh(attn(cat((s, encoder output))
	Size	[1, 1, 300]	[batch size, 1, 300]
	Name	a	c
	Formula	softmax(v(energy))	a · encoder output
	Size	[batch size, 1, 1]	[batch size, 1, 300]
	Output of Attention Layer		
	Name	decoder input	decoder hidden
	Formula	cat((decoder input, c)	s
	Size	[batch size, 1, 600]	[1, batch size, 300]
Decoder			

Table 4.7: Attention layer architeture

The encoder input is the labels of N pitches and rolls. The decoder input is the final hidden state of the encoder, which will then concatenated with c calculated in the attention layer. The decoder output will be concatenated with c and the encoder output. The final output can be obtained after a linear layer.

4.5.3 GRU encoder attention decoder PR model

This model is similar to LSTM encoder attention decoder PR model, the only difference is that it's the GRU module being used in this model, precise illustration of the architecture can be seen in **A.1**.

Input	Sequence of N pitches and rolls [$1 \times \mathbf{N} \cdot 2$]					
Name	*Layer*	*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	GRU	1	Ture	1×300	$1 \times N \cdot 2$	1×300
Attention	Tanh + FC				1×600	1×300
v	Soft + FC				1×300	1×1
Decoder	Relu + GRU	1	True	1×600	1×600	1×300
FC 1	FC + ReLu				1×600	1×150
Output	FC + Tanh				1×150	$1 \times M \cdot 2$

Table 4.8: GRU encoder attention decoder PR model architecture

4.5.4 CNN GRU encoder decoder images PR model

This model is similar to CNN LSTM encoder decoder images PR model in [8], the only difference is that it's the GRU module being used in this model, precise illustration of the architecture can be seen in **A.1**.

Non-confidential report and publishable on the Internet

Input	Sequence of N frames [54 × 96 × 3]						
N CNN parts							
Name	*Layer*	*Number filters*	*Filter size*	*Stride*	*Padding*	*Inpout size*	*Output size*
Conv 1	Convolution ReLu	8 -	3×3 -	1 -	2 -	54 × 96 × 3 -	54 × 96 × 8 -
Pooling 1	Max pooling	1	3×3	2	1	54 × 96 × 8	27 × 48 × 8
Conv 2	Convolution ReLu	16 -	3×3 -	1 -	1 -	27 × 48 × 8 -	27 × 48 × 16 -
Pooling 2	Max Pooling	1	3×3	2	1	27 × 48 × 16	14 × 24 × 16
Conv 3	Convolution ReLu	32 -	3×3 -	1 -	1 -	14 × 24 × 16 -	14 × 24 × 32 -
Pooling 4	Max Pooling	1	3×3	2	1	14 × 24 × 32	7 × 24 × 32
FC Mu	FC + ReLu					1 × 2688	1 × 1024
FC Sigma	FC + Tanh					1 × 2688	1 × 1024
RNN Part: Adding sequence of N pitches and rolls $[N \times 2]$							
Name	*Layer*		*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	GRU		1	Ture	1 × 300	$1 \times \mathrm{N} \cdot 1024 + 2$	1 × 300
Decoder	Relu + GRU		1	True	1 × 300	1 × 300	1 × 300
FC 1	FC + ReLu					1 × 300	1 × 150
FC 2	FC + Tanh					1 × 150	1 ×2
Output							$1 \times M \cdot 2$

Table 4.9: CNN GRU encoder decoder PR model architecture

4.5.5 CNN LSTM encoder attention decoder images PR model

This model is similar to CNN LSTM encoder decoder images PR model in [8], an attention model is added to this model, precise illustration of the architecture can be seen in **A.1**.

Input	Sequence of N frames [54 × 96 × 3]						
N CNN parts							
Name	*Layer*	*Number filters*	*Filter size*	*Stride*	*Padding*	*Inpout size*	*Output size*
Conv 1	Convolution ReLu	8 -	3×3 -	1 -	2 -	54 × 96 × 3 -	54 × 96 × 8 -
Pooling 1	Max pooling	1	3×3	2	1	54 × 96 × 8	27 × 48 × 8
Conv 2	Convolution ReLu	16 -	3×3 -	1 -	1 -	27 × 48 × 8 -	27 × 48 × 16 -
Pooling 2	Max Pooling	1	3×3	2	1	27 × 48 × 16	14 × 24 × 16
Conv 3	Convolution ReLu	32 -	3×3 -	1 -	1 -	14 × 24 × 16 -	14 × 24 × 32 -
Pooling 4	Max Pooling	1	3×3	2	1	14 × 24 × 32	7 × 24 × 32
FC Mu	FC + ReLu					1 × 2688	1 × 1024
FC Sigma	FC + Tanh					1 × 2688	1 × 1024
RNN Part: Adding sequence of N pitches and rolls $[N \times 2]$							
Name	*Layer*		*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	GRU		1	Ture	1 × 300	$1 \times \mathrm{N} \cdot 1024 + 2$	1 × 300
Attention	Tanh + FC					1 × 600	1 × 300
v	Soft + FC					1 × 300	1 × 1
Decoder	Relu + GRU		1	True	1 × 300	1 × 300	1 × 300
FC 1	FC + ReLu					1 × 300	1 × 150
FC 2	FC + Tanh					1 × 150	1 ×2
Output							$1 \times M \cdot 2$

Table 4.10: CNN LSTM encoder attention decoder images PR model

4.5.6 CNN GRU encoder decoder images PR model

This model is similar to CNN LSTM encoder decoder images PR model in [8], the only difference is that it's the GRU module being used in this model, precise illustration of the architecture can be seen in **A.1**.

Input	Sequence of N frames [54 × 96 × 3]						
N CNN parts							
Name	*Layer*	*Number filters*	*Filter size*	*Stride*	*Padding*	*Inpout size*	*Output size*
Conv 1	Convolution ReLu	8 -	3×3 -	1 -	2 -	54 × 96 × 3 -	54 × 96 × 8 -
Pooling 1	Max pooling	1	3×3	2	1	54 × 96 × 8	27 × 48 × 8
Conv 2	Convolution ReLu	16 -	3×3 -	1 -	1 -	27 × 48 × 8 -	27 × 48 × 16 -
Pooling 2	Max Pooling	1	3×3	2	1	27 × 48 × 16	14 × 24 × 16
Conv 3	Convolution ReLu	32 -	3×3 -	1 -	1 -	14 × 24 × 16 -	14 × 24 × 32 -
Pooling 4	Max Pooling	1	3×3	2	1	14 × 24 × 32	7 × 24 × 32
FC Mu	FC + ReLu					1 × 2688	1 × 1024
FC Sigma	FC + Tanh					1 × 2688	1 × 1024
RNN Part: Adding sequence of N pitches and rolls $[N \times 2]$							
Name	*Layer*		*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	LSTM		1	Ture	1 × 300	$1 \times \mathrm{N} \cdot 1024 + 2$	1 × 300
Decoder	Relu + LSTM		1	True	1 × 300	1 × 300	1 × 300
FC 1	FC + ReLu					1 × 300	1 × 150
FC 2	FC + Tanh					1 × 150	1 ×2
Output							$1 \times M \cdot 2$

Table 4.11: CNN LSTM encoder decoder PR* architecture

4.5.7 CNN LSTM encoder GRU attention decoder images PR model

This model is similar to CNN LSTM encoder attention decoder images PR model in [8], the only difference is that LSTM is replaced by GRU, precise illustration of the architecture can be seen in **A.1**.

Input	Sequence of N frames [54 × 96 × 3]						
N CNN parts							
Name	*Layer*	*Number filters*	*Filter size*	*Stride*	*Padding*	*Inpout size*	*Output size*
Conv 1	Convolution ReLu	8 -	3×3 -	1 -	2 -	54 × 96 × 3 -	54 × 96 × 8 -
Pooling 1	Max pooling	1	3×3	2	1	54 × 96 × 8	27 × 48 × 8
Conv 2	Convolution ReLu	16 -	3×3 -	1 -	1 -	27 × 48 × 8 -	27 × 48 × 16 -
Pooling 2	Max Pooling	1	3×3	2	1	27 × 48 × 16	14 × 24 × 16
Conv 3	Convolution ReLu	32 -	3×3 -	1 -	1 -	14 × 24 × 16 -	14 × 24 × 32 -
Pooling 4	Max Pooling	1	3×3	2	1	14 × 24 × 32	7 × 24 × 32
FC Mu	FC + ReLu					1 × 2688	1 × 1024
FC Sigma	FC + Tanh					1 × 2688	1 × 1024
RNN Part: Adding sequence of N pitches and rolls $[N \times 2]$							
Name	*Layer*		*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	LSTM		1	Ture	1 × 300	$1 \times \mathrm{N} \cdot 1024 + 2$	1 × 300
Attention	Tanh + FC					1 × 600	1 × 300
v	Soft + FC					1 × 300	1 × 1
Decoder	Relu + GRU		1	True	1 × 300	1 × 300	1 × 300
FC 1	FC + ReLu					1 × 300	1 × 150
FC 2	FC + Tanh					1 × 150	1 ×2
Output							1 × M ·2

Table 4.12: CNN LSTM encoder GRU attention decoder images PR model

4.6 Models with sequence length of size N

In this section, models with sequence length of size N will be presented. Here size N means of the type $[batchsize, \mathbf{N}, *]$, where $*$ can be replaced by any number. The encoder output is the ensemble of the hidden states of all time steps. In the model with sequence length of size 1, there is a only one hidden state of only one time step, which

may not make full use of the Attention Mechanism, so a change is made to the structure of the models. For detailed differnece between these two types of models, see **4.4**, precise illustration of the architecture can be seen in **A.2**.

4.6.1 LSTM encoder decoder PR*

In this model, the shape of the input of LSTM is firstly changed from $[batchsize, 1, 2 \cdot N]$ to $[batchsize, N, 2]$, where 2 means the value of pitch and roll. Just like the [LSTM encoder decoder PR] model, the decoder output will directly be the decoder input, the initial hidden states will be zero tensors. The architecture of the new model can be seen in the table below, precise illustration of the architecture can be seen in **A.2**.

Input	Sequence of N pitches and rolls $[N \times 2]$					
Name	*Layer*	*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	LSTM	1	Ture	1 × 300	N × 2	N × 300
Decoder	Relu + LSTM	1	True	1 × 300	N × 300	N × 300
FC 1	FC + ReLu				N × 300	N × 150
Output	FC + Tanh				N × 150	N ×2

Table 4.13: LSTM encoder decoder PR model architecture version 1

Even though the sequence length of this model is no longer 1, the input length is always equal to that of the output length, which is not suitable when the data are not enough. In cosideration of this, we changed the models according to the method in this blog[9] in order to have a LSTM model with sequence length of size N, precise illustration of the architecture can be seen in **A.2**.

In this model, a sequence of N pitches and rolls is taken as the input of the LSTM encoder, then this time not the whole encoder output but only the last hidden state of the encoder as well as the last state of the encoder input will be used as the initial input of the LSTM decoder. Then for a cycle of M times, the decoder output of the last time step will be used as the input of decoder for the next time step. The architecture can be seen in the table below. The initial hidden states of both the encoder and the decoder are still zero tensors. The output of all the decoders will be concatenate into one tensor as the result of prediction.

Input	Sequence of N pitches and rolls $[N \times 2]$					
Name	*Layer*	*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	LSTM	1	Ture	1 × 300	N × 2	N × 300
Encoder hidden state as decoder hidden state, last encoder input as initial decoder input						
Decoder	Relu + LSTM	1	True	1 × 300	1 × 300	1 × 300
FC 1	FC + ReLu				1 × 300	1 × 150
FC 2	FC + Tanh				1 × 150	1 ×2
Number of Decoder, FC 1, FC 2 = M, decoder output as new deocder input of next cycle						
Output						M ×2

Table 4.14: LSTM encoder decoder PR* architecture

[9] https://jeddy92.github.io/JEddy92.github.io/ts_seq2seq_intro/

4.6.2 GRU encoder decoder PR*

This is a model similar to LSTM encoder decoder PR many, the only difference is that the model used is GRU. The architecture can be seen from the table below, precise illustration of the architecture can be seen in **A.2**.

Input	Sequence of N pitches and rolls [$N \times 2$]					
Name	*Layer*	*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	GRU	1	Ture	1 × 300	N × 2	N × 300
Decoder	Relu + GRU	1	True	1 × 300	1 × 300	1 × 300
Encoder hidden state as decoder hidden state, last encoder input as initial decoder input						
Decoder	Relu + LSTM	1	True	1 × 300	1 × 300	1 × 300
FC 1	FC + ReLu				1 × 300	1 × 150
FC 2	FC + Tanh				1 × 150	1 ×2
Number of Decoder, FC 1, FC 2 = M, decoder output as new deocder input of next cycle						
Output						M ×2

Table 4.15: GRU encoder decoder PR* architecture

4.6.3 LSTM encoder attention decoder PR*

In this section, the attention layer is added to the model with sequence length of size N, precise illustration of the architecture can be seen in **A.2**.

Input	Sequence of N pitches and rolls [$N \times 2$]					
Name	*Layer*	*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	LSTM	1	Ture	1 × 300	N × 2	N × 300
Attention	Tanh + FC				1 × 600	1 × 300
v	Soft + FC				1 × 300	1 × 1
Decoder	Relu + LSTM	1	True	1 × 300	1 × 300	1 × 300
FC 1	FC + ReLu				1 × 300	1 × 150
FC 2	FC + Tanh				1 × 150	1 ×2
Number of Decoder, FC 1, FC 2 = M, decoder output as new deocder input of next cycle						
Output						M ×2

Table 4.16: LSTM encoder attention decoder PR* architecture

4.6.4 GRU encoder attention decoder PR*

This model is similar to LSTM encoder attention decoder PR*, the only difference is that it's the GRU module bing used in this model, precise illustration of the architecture can be seen in **A.2**.

4.6.5 LSTM encoder GRU attention decoder PR*

In this model, the GRU is used as a complementary unit of the LSTM. An encoder of LSTM and a decoder of GRU is used. The attention architecture is the same as that of table 4.8. The architecture is shown here, precise illustration of the architecture can be seen in **A.2**.

Non-confidential report and publishable on the Internet

Input	Sequence of N pitches and rolls $[N \times 2]$					
Name	*Layer*	*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	LSTM	1	Ture	1×300	$N \times 2$	$N \times 300$
Attention	Tanh + FC				1×600	1×300
v	Soft + FC				1×300	1×1
Decoder	Relu + GRU	1	True	1×300	1×300	1×300
FC 1	FC + ReLu				1×300	1×150
FC 2	FC + Tanh				1×150	1×2
Number of Decoder, FC 1, FC 2 = M, decoder output as new deocder input of next cycle						
Output						$M \times 2$

Table 4.17: LSTM encoder GRU attention decoder PR* architecture

4.6.6 CNN LSTM encoder decoder images PR*

This is a model based on the previous [CNN LSTM encoder decoder images PR] [8], the model is consist of two parts: a CNN part processing the pictures, whose result will then be concatenated with the labels of pitch and roll, and a RNN part processing the new tensor. The architecture of the new model is presented in the table below, precise illustration of the architecture can be seen in **A.2**.

Input	Sequence of N frames $[54 \times 96 \times 3]$						
N CNN parts							
Name	*Layer*	*Number filters*	*Filter size*	*Stride*	*Padding*	*Inpout size*	*Output size*
Conv 1	Convolution ReLu	8 -	3×3 -	1 -	2 -	$54 \times 96 \times 3$ -	$54 \times 96 \times 8$ -
Pooling 1	Max pooling	1	3×3	2	1	$54 \times 96 \times 8$	$27 \times 48 \times 8$
Conv 2	Convolution ReLu	16 -	3×3 -	1 -	1 -	$27 \times 48 \times 8$ -	$27 \times 48 \times 16$ -
Pooling 2	Max Pooling	1	3×3	2	1	$27 \times 48 \times 16$	$14 \times 24 \times 16$
Conv 3	Convolution ReLu	32 -	3×3 -	1 -	1 -	$14 \times 24 \times 16$ -	$14 \times 24 \times 32$ -
Pooling 4	Max Pooling	1	3×3	2	1	$14 \times 24 \times 32$	$7 \times 24 \times 32$
FC Mu	FC + ReLu					1×2688	1×1024
FC Sigma	FC + Tanh					1×2688	1×1024
RNN Part: Adding sequence of N pitches and rolls $[1 \times N \cdot 2]$							
Name	*Layer*		*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	LSTM		1	Ture	1×300	$N \times 1024 + 2$	$N \times 300$
Decoder	Relu + LSTM		1	True	1×300	1×300	1×300
FC 1	FC + ReLu					1×300	1×150
FC 2	FC + Tanh					1×150	1×2
Number of Decoder, FC 1, FC 2 = M, decoder output as new deocder input of next cycle							
Output							$M \times 2$

Table 4.18: CNN LSTM encoder decoder images PR*

4.6.7 CNN LSTM encoder attention decoder images PR*

In this model, an attention model is added. The architecture of the new model is presented in the table below, precise illustration of the architecture can be seen in **A.2**.

Input	Sequence of N frames [54 × 96 × 3]						
N CNN parts							
Name	*Layer*	*Number filters*	*Filter size*	*Stride*	*Padding*	*Inpout size*	*Output size*
Conv 1	Convolution ReLu	8 -	3×3 -	1 -	2 -	54 × 96 × 3 -	54 × 96 × 8 -
Pooling 1	Max pooling	1	3×3	2	1	54 × 96 × 8	27 × 48 × 8
Conv 2	Convolution ReLu	16 -	3×3 -	1 -	1 -	27 × 48 × 8 -	27 × 48 × 16 -
Pooling 2	Max Pooling	1	3×3	2	1	27 × 48 × 16	14 × 24 × 16
Conv 3	Convolution ReLu	32 -	3×3 -	1 -	1 -	14 × 24 × 16 -	14 × 24 × 32 -
Pooling 4	Max Pooling	1	3×3	2	1	14 × 24 × 32	7 × 24 × 32
FC Mu	FC + ReLu					1 × 2688	1 × 1024
FC Sigma	FC + Tanh					1 × 2688	1 × 1024
RNN Part: Adding sequence of N pitches and rolls [$N \times 2$]							
Name	*Layer*		*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	LSTM		1	Ture	1 × 300	N × 1024 + 2	N × 300
Attention	Tanh + FC					1 × 600	1 × 300
v	Soft + FC					1 × 300	1 × 1
Decoder	Relu + GRU		1	True	1 × 300	1 × 300	1 × 300
FC 1	FC + ReLu					1 × 300	1 × 150
FC 2	FC + Tanh					1 × 150	1 ×2
Number of Decoder, FC 1, FC 2 = M, decoder output as new deocder input of next cycle							
Output							M ×2

Table 4.19: CNN LSTM encoder decoder PR* architecture

4.6.8 CNN LSTM encoder GRU attention decoder images PR*

In this model, a LSTM is used as the encoder and a GRU is used as the decoder, whose architecture is shown below, precise illustration of the architecture can be seen in **A.2**.

Input	Sequence of N frames [54 × 96 × 3]						
N CNN parts							
Name	*Layer*	*Number filters*	*Filter size*	*Stride*	*Padding*	*Inpout size*	*Output size*
Conv 1	Convolution ReLu	8 -	3×3 -	1 -	2 -	54 × 96 × 3 -	54 × 96 × 8 -
Pooling 1	Max pooling	1	3×3	2	1	54 × 96 × 8	27 × 48 × 8
Conv 2	Convolution ReLu	16 -	3×3 -	1 -	1 -	27 × 48 × 8 -	27 × 48 × 16 -
Pooling 2	Max Pooling	1	3×3	2	1	27 × 48 × 16	14 × 24 × 16
Conv 3	Convolution ReLu	32 -	3×3 -	1 -	1 -	14 × 24 × 16 -	14 × 24 × 32 -
Pooling 4	Max Pooling	1	3×3	2	1	14 × 24 × 32	7 × 24 × 32
FC Mu	FC + ReLu					1 × 2688	1 × 1024
FC Sigma	FC + Tanh					1 × 2688	1 × 1024
RNN Part: Adding sequence of N pitches and rolls [$N \times 2$]							
Name	*Layer*		*Number Layers*	*Batch First*	*Hidden size*	*Input size*	*Output size*
Encoder	LSTM		1	Ture	1 × 300	N × 1024 + 2	N × 300
Decoder	Relu + GRU		1	True	1 × 300	1 × 300	1 × 300
FC 1	FC + ReLu					1 × 300	1 × 150
FC 2	FC + Tanh					1 × 150	1 ×2
Number of Decoder, FC 1, FC 2 = M, decoder output as new deocder input of next cycle							
Output							M ×2

Table 4.20: CNN LSTM encoder GRU decoder images PR* architecture

4.7 TransformerModel PR model

In this section, a model without the direct use of RNN will be presented. This model is created based on the the transformer model [14]. Thanks to the nn.transformer

module of Pytorch, we can create this model without the recreating all the layers. We have created one transformer encoder and one transformer decoder.

The encoder input is a sequence of length N, with each part of the sequence is consist of one value and pitch and one value of roll. The decoder input is a sequence of length M. Before going to the Transformer model, both the encoder input and the decoder input will be processed by the position encoding layer, which will add a certain value to the input to create the relationship of the position of each value in the sequence.

Encoder Input	Sequence of N pitches and rolls [1 × 2]					
Name	*Layer*	*Number Layers*	*Batch first*	*HIdden size*	*Input Size*	*Output size*
Pos encoder	Positional Encoding	1	-	-	N × 2	N × 2
Transformer Encoder	MultiAtten: Linear	1	-	-	N × 2	N × 2
	Linear	1	-	-	N × 2	N × 300
	Dropout	1	-	-	N × 300	N × 300
	Linear	1	-	-	N × 300	N × 2
	LayerNorm	1	-	-	N × 2	N × 2
	LayerNorm	1	-	-	N × 2	N × 2
	Dropout	1	-	-	N × 2	N × 2
	Dropout	1	-	-	N × 2	N × 2
Encoder Input	Sequence of M pitches and rolls [1 × 2]					
Pos encoder	Positional Encoding	1	-	-	M × 2	M × 2
Transformer Decoder	SelfAtten: linear	1	-	-	M × 2	M × 2
	MultiAtten: linear	1	-	-	M × 2	M × 2
	Linear	1	-	-	M × 2	M × 300
	Dropout	1	-	-	M × 300	M × 300
	Linear	1	-	-	M × 300	M × 2
	LayerNorm	1	-	-	M × 2	M × 2
	LayerNorm	1	-	-	M × 2	M × 2
	Dropout	1	-	-	M × 2	M × 2
	Dropout	1	-	-	M × 2	M × 2
Output:	Dropout	1	-	-	M × 2	M × 2

Table 4.21: TransformerModel PR model architecture

Non-confidential report and publishable on the Internet

5 Model Training

The implementation of the models is finished in help of PyTorch[10]. PyTorch is an open source machine learning library based on the Torch library, used for applications such as computer vision and natural language processing. Compared with Tenserflow, Pytorch is much more friendly and easy-going. The time series forecasting problem in this internship belongs to the class of supervised learning since the relation between the input and the output is already known. It belongs also to the regression problem because the input and the output are continuous values.

5.1 Parameters

Before training, some parameters of the models should be introduced. Thanks to work of Nazar [8], the structure of the software has already been created in the form of bash, where some parameters have been created. The main parameters in this models can be seen in the table below:

Parameters	Explanation
Past window size	The number of seconds of simulation taken to predict pitch and roll (sequence of frames)
Future window size	the number of seconds for which pitch and roll will be predicted (sequence of frames)
Number of episodes	The number of episodes used as input data
Latent vectors	The dimensions of the latent vectors (hidden size) for the LSTM or GRU parts of the model
Weight decay	A hyperparameter which consists of adding a penalty to the error function that depends on the magnitude of the weights that connect neurons to each other. A regularization technique used to limit overfitting in a neural network[11].
Batchsize	a hyperparameter that defines the number of samples to work through before updating the internal model parameters
Number of Epochs	A hyperparameter that defines the number of times that the learning algorithm will work through the entire training dataset
Frames per second (fps)	A data generation parameter. More fps is needed when the input is a video, in our case with the labels and images as input, 1 or two is enough. And the difference between the choice of 1 or 2 can be detected.

Table 5.1: Models nomenclature for each module component

5.2 Data loading

For machine learning, the process of preparing the data is very important. Thanks to the data generation scripts, we have created over 600 episodes of data, with each of them containing 400 consecutive photos. Besides, the data has been created with increasing

[10] https://pytorch.org/

Non-confidential report and publishable on the Internet

complexity of the sea state (wave height, the wing speed, the scale, etc.). The complexity of the sea state helps improve the variety of the data used to train.

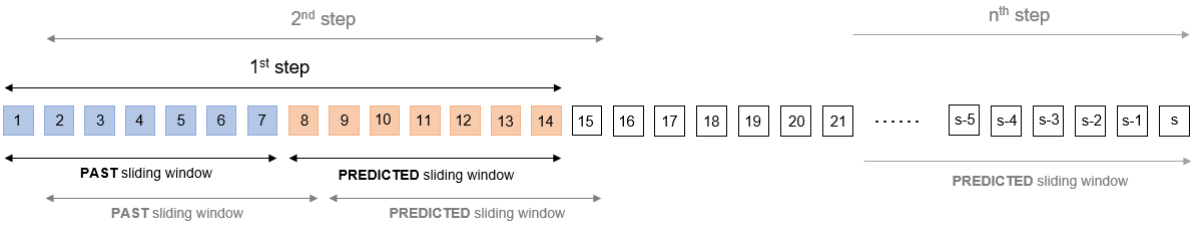


Figure 5.1: Illustration of data iteration step by step for models with sequence length of size 1

The length of the sequence of input data will be of size [60, 72, 120] frames, which is decided by the specific use of the LSTM model. The passage through this sequence will be carried out step by step (see Figure 5.1), where Blue images mean past sliding window, red images mean predicted sliding window. The blue images and the red images together is one data loader through the sequence.

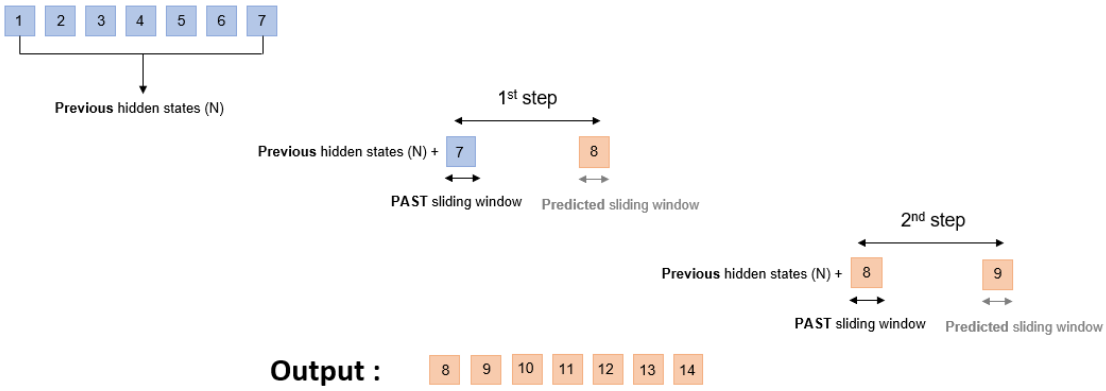


Figure 5.2: Illustration of data iteration step by step for models with sequence length of size N

The encoder output is the ensemble of the hidden states of all time steps. In the model with sequence length of size 1, there is a only one hidden state of only one time step, and it's used only once. It can be seen in **Figure 5.2** that, in the model with sequence length of size N, the hidden states are used and updated in each cycle of predicting a new frame, which caters to the requirements of the Attention Mechanism.

The data generated has been separated into three parts: training data, validating data and test data, which ensure the independence among the three data set.

5.3 Loss function and Optimizer

Mean Square Error (MSE), one of the most commonly used regression loss function is used in this internship to be loss function. The objective of the training is to gain the minimum of loss function.

MSE is the sum of squared distances between our target variable and predicted values (here the predicted angle and the original simulated ground. Since we predict the angles for a sequence of frames, the error is assumed for the whole sequence.

$$loss = MES = \frac{\sum_{i=1}^{number\ of\ frames}(Predict - Original)^2}{number\ of\ frames}$$

When building the neural network model, the best optimizer should be chosen to brake fast convergence and learn correctly, while adjusting internal parameters to minimize the loss function. In this internship, Adam is used as the optimizer. Adam [22] is a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement, showing huge performance gains in terms of speed of training.

5.4 Underfitting and Overfitting problem

Underfitting usually happens when the model is too simple. The root cause of the under-fitting problem is that the feature dimension is too small, which causes the fitted function to fail to satisfy the training set and error penetration. This can usually be solved by introducing complex models such as polynomials or neural networks.

Overfitting usually means that the model learned by training has poor performance on the test set, but very good performance on the training set, that is, overfitting on the training set, with poor generalization ability. The root cause of the over-fitting problem is that there are too many feature dimensions, which leads to the perfect function of the fit through the training set, but the prediction result of the new data is poor. In order to avoid the problem of overfitting, a method of early stopping has been introduced in the previous model, which means that the training will be stopped automatically when the training loss can no longer be reduced by training.

5.5 Hyperparameters adjustment

Setting system hyperparameters is one of the most important and complex aspects of deep learning, the right parameters can help us to achieve the best effect of the models. However it takes a lot of work to find the best parameters.

There are many methods to find the best parameters, but all of them take a lot of time and computing power. In this internship, the hyperband algorithm, which is a mixture of grid search and random search is used, which can help us to find the best hyperparameters automatically.

Non-confidential report and publishable on the Internet

6 Model Testing

In this chapter, several experiments on testing the models already created are carried out and the results will be presented. A basic setting is firstly used to test all the models, then for the models having good results, the Hyperband algorithm will be used to find the best hyper-parameters.

hyper-parameters	
Number of episodes	670
Number of epochs	60
Learning rate	1 e-04
Weight decay	1 e-03
Frame per second (fps)	2
Past window size	10 s
Future window size	12 s

Table 6.1: Basic hyper-parameters for models

The models created in this internship are separated into three groups: models with only one time step, models with only many time steps and the model without the direct use of RNN model. The models will firstly be compared in each group and then the best models of each group will be compared to find the model which has the best performance.

6.1 Models with sequence length of size 1

In this section attention is paid to the models sequence length of size 1. The models will be separated into two pars: those with a CNN part and those without a CNN part. The models are tested with the hyper-parameters in **Table 6.1**.

6.1.1 Models without a CNN part

In this section the following models will be tested with the learning rate $1e^{-3}$: [**GRU encoder decoder PR**], [**GRU encoder attention decoder PR**], [**LSTM encoder attention decoder PR**].

Non-confidential report and publishable on the Internet

MODELS	MSE train $\times 10^{-2}$ avg over sum pitch and roll	MSE val $\times 10^{-2}$ avg over sum pitch and roll	MSE test $\times 10^{-2}$ avg over sum pitch and roll	Early Stopping
GRU encoder decoder PR	0.44923679	0.48331361	0.50718301	No
GRU encoder attention decoder PR	0.55111838	0.59610689	0.62536183	No
LSTM encoder decoder PR [baseline]	0.71639374	0.72111610	0.78574833	No
LSTM encoder attention decoder PR	0.54478331	0.58919667	0.62397669	No

Table 6.2: Results for models with only one time step but without a CNN part. *Red line* - our baseline [LSTM encoder decoder PR] model, worst result; *Green line* - the best result at the moment

Results in **Table 6.2** show that the GRU models perform better than LSTM models when the encoder output is of size 1, it proves that GRU has better behaviour than LSTM for some certain small data set where GRU models converge much more quickly than LSTM models. Besides, the effect of the Attention mechanism has also been verified. It can be found that when added a layer of attention, the LSTM models perform better and achieve lower loss. However, the attention mechanism dosn't have a good performance in the GRU models, which can be explained by the restriction of the size of the hidden state of only one time step.

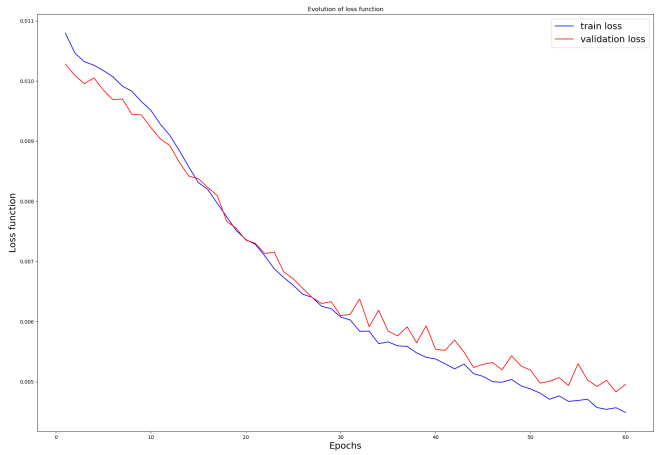


Figure 6.1: Loss function for [GRU encoder decoder PR] model

From the figure of loss function, it can be seen that both the train loss and the validation loss are reduced by training, however, the incorrect value of the learning rate is shown by the gradient of the line.

6.1.2 Models with a CNN part

In this section the following models will be tested : **[CNN GRU encoder decoder images PR]**, **[CNN GRU encoder attention decoder images PR]**, **[CNN LSTM**

encoder attention decoder images PR] and [**CNN LSTM encoder GRU attention decoder images PR**],

MODELS	MSE train $\times 10^{-2}$ avg over sum pitch and roll	MSE val $\times 10^{-2}$ avg over sum pitch and roll	MSE test $\times 10^{-2}$ avg over sum pitch and roll	Early Stopping
CNN LSTM encoder decoder images PR [baseline]	0.04634282	0.09767304	0.11058501	No
CNN GRU encoder decoder images PR	0.05409161	0.10109522	0.11552966	No
CNN LSTM encoder attention decoder images PR	0.04790248	0.10190074	0.12174388	No
CNN GRU encoder attention decoder images PR	0.06624670	0.12175015	0.13829688	No
CNN LSTM encoder GRU attention decoder images PR	0.05527550	0.11168915	0.13224233	Yes

Table 6.3: Results for models with sequence length of size 1 and with a CNN part. *Red line* - our baseline [CNN LSTM encoder decoder images PR] model, *Green line* [CNN GRU encoder decoder images PR] model, best model created

From the result, we can find that although GRU has better performance when there is no CNN part, it's still the LSTM model who performs better when the input data become more complex, which corresponds to the characteristic of GRU.

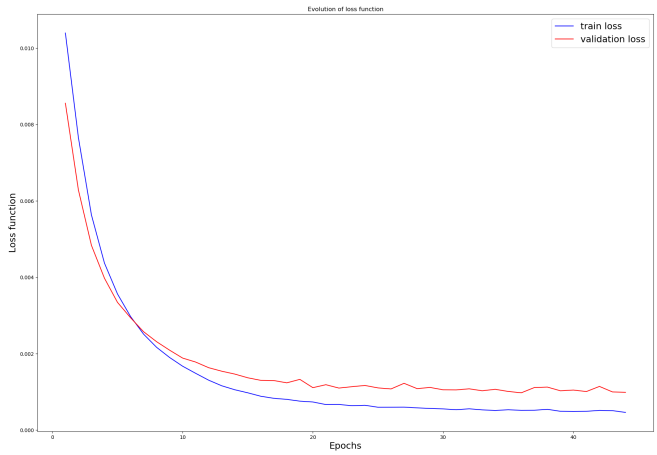


Figure 6.2: Loss function for [CNN LSTM encoder decoder images PR] model

The results also show that the effect of the Attention Mechanism in one-sequence-length model is not obvious. No matter in a model of LSTM or GRU, the models perform worse when an attention layer is added. The Early Stopping is triggered during the training of the model [CNN LSTM encoder GRU attention decoder images PR], which means that the hyper-parameters should be adjusted.

The figure of the evolution of loss function shows the reduction of both the train loss and the validation loss during the training. However an undulation can be seen in the

end of the training, which is caused by the too long training time. Even though there is a script of early-stopping helping us to resolve the problem of overfitting, attention still should be paid on the adjustment of training time and learning rate.

6.2 Models with sequence length of size N

In this section attention is paid to the models with sequence length of size N, which means the encoder output size is of the type [*batchsize*, **N**, $*$], where $*$ can be replaced by any number. The output of a RNN module is an ensemble of the hidden states of all time steps, so in these models, hidden states of many time steps can be obtained, which caters to the requirements of the Attention Mechanism.

MODELS	MSE train $\times 10^{-2}$ avg over sum pitch and roll	MSE val $\times 10^{-2}$ avg over sum pitch and roll	MSE test $\times 10^{-2}$ avg over sum pitch and roll	Early Stopping
LSTM encoder decoder PR*	1.03903012	0.99447452	1.11595568	Yes
GRU encoder decoder PR* PR	1.03707045	0.99144033	1.11294128	Yes
LSTM encoder attention decoder PR*	1.04145141	0.99824596	1.11859753	Yes
LSTM encoder GRU attention decoder PR*	1.04472465	1.00621673	1.12370137	Yes
GRU encoder attention decoder PR*	1.04472465	1.00621673	1.12370137	Yes
CNN LSTM encoder decoder images PR*	0.18351872	0.23218019	0.26445403	No
CNN LSTM encoder attention decoder images PR*	0.09051899	0.13835708	0.16287565	No
CNN LSTM encoder GRU attention decoder images PR*	0.09346003	0.14311764	0.16784911	Yes

Table 6.4: Results for models with many time steps. *Red line* - [LSTM encoder GRU attention decoder PR*] model, worst model. *Green line* - [CNN LSTM encoder attention decoder images PR] model, best model.

The results show that the Early Stopping operation is triggered by most of the models with many time steps without a CNN part, which caused by the underfitting problem. For a complex model of this kind, the number of features of the input of each time step is only 2 (pitch and roll), which causes the underfitting. It doesn't help by changing the optimizer so it may be a problem of initial hyper-parameters. The hyperband algorithm can help us to find proper hyper-parameters. The better performance of models with a CNN part has proved the effect of the CNN module, which also increases the depth of the model. The input for models with CNN is the images concatenated with the labels, so there is 1026 features for each time step, which helps to resolve the problem of underfitting.

Non-confidential report and publishable on the Internet

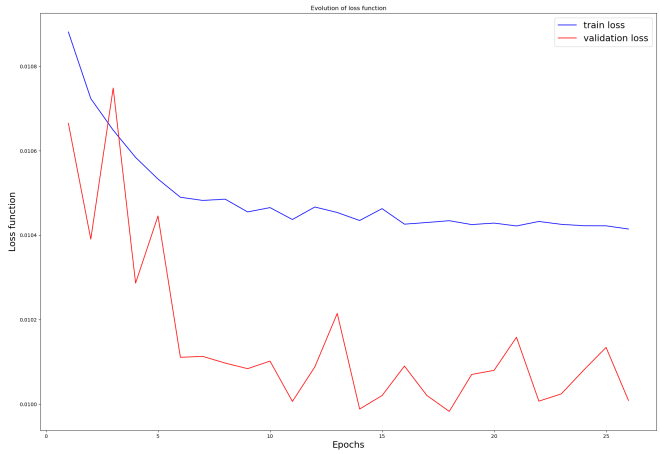


Figure 6.3: Loss function for [LSTM encoder decoder images PR*] model

Because of the early stopping operation, the advantage of the Attention Mechanism is not shown in the models without a CNN part. But the GRU model doesn't perform better with an attention layer when there is no CNN part, which agrees with the the result of models before. And for the models with a CNN part, the model with am attention mechanism performs much better than that without an attention layer, the problem of long term dependencies is solved here.

Both in models with sequence length of size 1 and of size N, when there is both a GRU unit and a LSTM unit, the model doesn't get any better.

From the figure of the evolution of loss function of [LSTM encoder decoder images PR*] model, the fluctuation of both the train loss and validation loss during the training can be seen, which shows clearly the underfitting problem.

6.3 Model without direct use of RNN model

In this section the model without the direct use of RNN model: [Transformer-Model_PR] is tested wit the hyper-parameters

MODELS	MSE train $\times 10^{-2}$ avg over sum pitch and roll	MSE val $\times 10^{-2}$ avg over sum pitch and roll	MSE test $\times 10^{-2}$ avg over sum pitch and roll	Early Stopping
TransformerModel PR	1.30421771	1.08398087	1.20638880	No

Table 6.5: Results for models without direct use of RNN.

6.4 Models comparison

After carrying out the tests of the model created, we now compare the best model of each group. The results can be found in the table below, which shows the normalized average MSE of the sum of pitch and roll over the predicted sequence (of 24 frames of length).

6.4.1 Comparison between the models with different sequence length

With the hyper-parameters in **Table 6.1**, we find that the results of the types of models are similar, most of the results of models with sequence length of size 1 are better than those of N, which can be explained by the complexity of the latter models.

The Attention Mechanism doesn't show good results in models with sequence length of size 1, the hidden state with only one time step can't not make full use of it. On the contrast, the models with sequence length of size N perform better when an attention layer is added and trained correctly, which proved our choice of changing the models.

6.4.2 Comparison between the models the best performance

In this section, a comparison will be made between the models of different types which perform the best.

MODELS	MSE train $\times 10^{-2}$ avg over sum pitch and roll	MSE val $\times 10^{-2}$ avg over sum pitch and roll	MSE test $\times 10^{-2}$ avg over sum pitch and roll	Early Stopping
LSTM encoder decoder PR [baseline]	0.71639374	0.72111610	0.78574833	No
GRU encoder decoder PR	0.44923679	0.48331361	0.50718301	No
CNN GRU encoder decoder images PR	0.05409161	0.10109522	0.11552966	No
CNN LSTM encoder attention decoder images PR*	0.09051899	0.13835708	0.16287565	No
TransformerModel PR	1.30421771	1.08398087	1.20638880	No

Table 6.6: Comparison of the best models in different type. *Red line* - [LSTM encoder decoder PR] model, our baseline model; *Strong green line* - [CNN GRU encoder attention decoder images PR] model, the best model at this moment; *Light green line* - [CNN LSTM encoder attention decoder images PR*] model, second best model at this moment.

For better understanding, the evolution of the loss function is also considered for different models during training on the training and validation data sets. The last plot also shows the MSE value on the test data set.

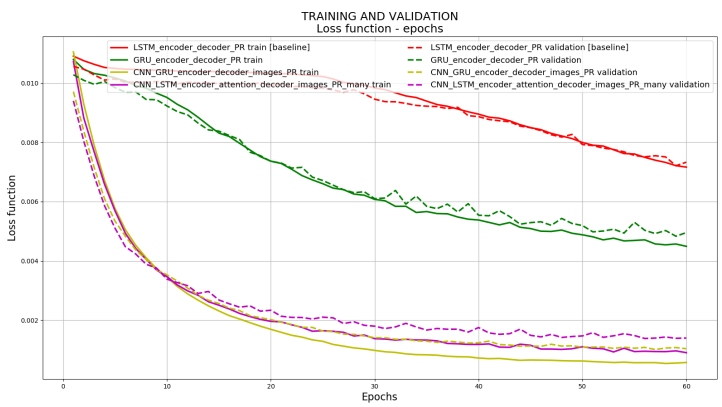


Figure 6.4: Loss function on the **training** and the **validation** data for all models

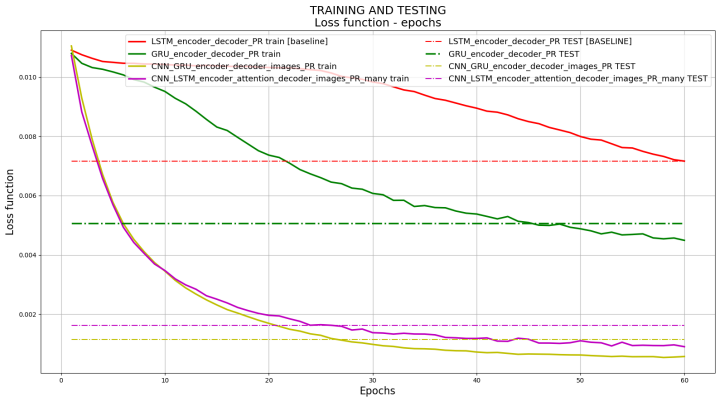


Figure 6.5: Loss function on the **training** data with **test** results the best models

We can explore further the components of our prediction - pitch and roll. Training results denormalized MSE values at the 10th predicted second can be seen in Table 6.7.

MODELS	MSE pitch at 10th sec [denormalized]	MSE roll at 10th sec [denormalized]
LSTM encoder decoder PR [baseline]	28.41066342071262	30.294543939312213
GRU encoder decoder PR	13.391954981220962	19.902441776554205
CNN GRU encoder decoder images PR	4.230484480904091	3.97595580751361
CNN LSTM encoder attention decoder images PR*	5.903502651304524	6.4243086576074155
TransformerModel PR	44.008785804647	34.688842885377184

Table 6.7: Comparison of the best models in different type. *Red line* - [LSTM encoder decoder PR] model, our baseline model; *Strong green line* - [CNN GRU encoder attention decoder images PR] model, the best model at this moment; *Light green line* - [CNN LSTM encoder attention decoder images PR*] model, second best modelat this moment.

As can be seen from the table, for a model with RNN and without CNN, the prediction of pitch angle is a more complex task, the error of this prediction is always larger than that for the roll angle. The condition is completely opposite for models with CNN,

We identified two models with the best performance in pitch and roll that we predict. **CNN GRU encoder decoder images PR** model and **CNN LSTM encoder attention decoder images PR*** model. Since the results are more balanced (there is no huge difference between pitch MSE and roll MSE for the model) , the work was mainly done to improve them.

Results of actual angles predictions of **LSTM encoder decoder PR model [baseline]** are in Figure 6.6, for **CNN GRU encoder decoder images PR** model in Figure 6.7 and for **CNN LSTM encoder attention decoder images PR*** model in Figure 6.8 .

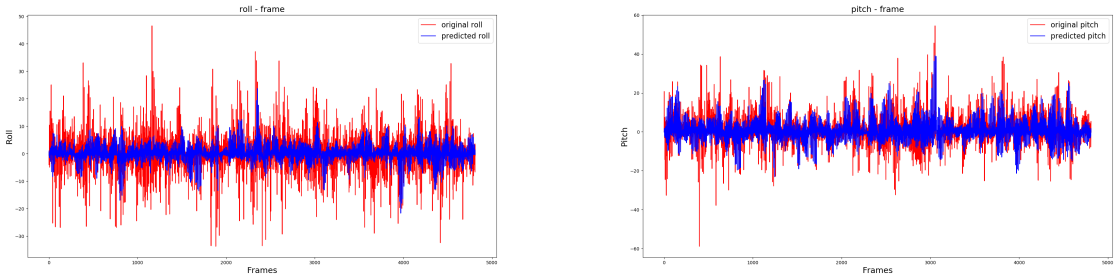


Figure 6.6: Original and predicted pitch (left) and roll (right) at 10th second using **LSTM encoder decoder PR** model; *Red*: original labels, *Blue*: Predicted data.

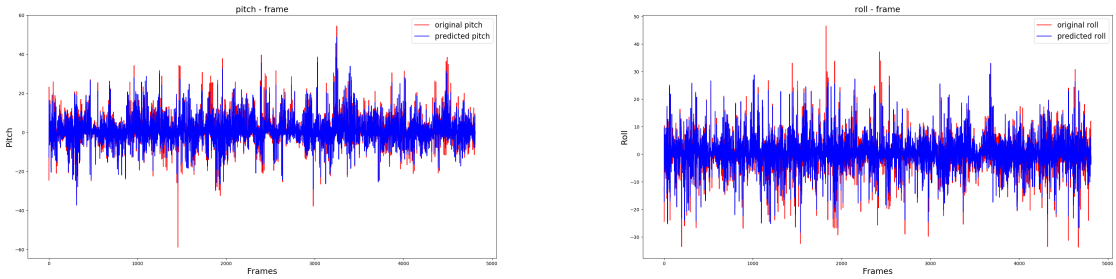


Figure 6.7: Original and predicted pitch (left) and roll (right) at 10th second using **CNN GRU encoder decoder images PR roll** model; *Red*: original labels, *Blue*: Predicted labels

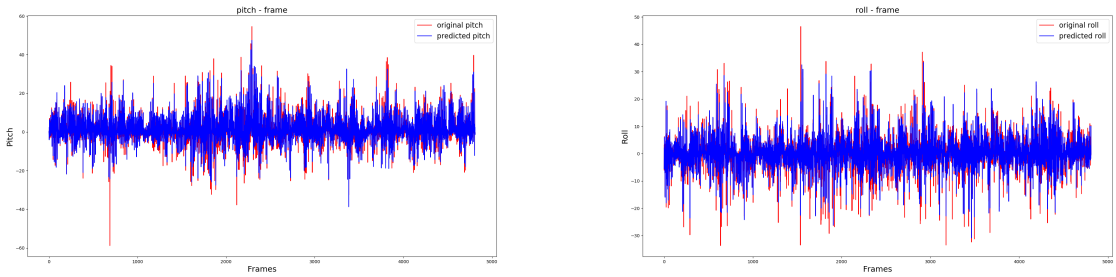


Figure 6.8: Original and predicted pitch (left) and roll (right) at 10th second using **CNN LSTM encoder attention decoder images PR*** model; *Red*: original labels, *Blue*: Predicted labels

As can be seen from the above figures, the results of those two models are significantly different. For models without CNN part, it's the roll which is better forecasted. By contrast, it's the value of pitch which is easier to predict for the models with CNN. Despite their good results, it is clear that there are peaks that are poorly predictable. So in the following section, the Hyperband algorithm will be used to find the best hyper-parameters for training.

6.5 Relation between the size of Past sliding window and Predicted sliding window

In this internship, in order to predict the ship motion in a longer future, we are using a smaller Past sliding window size than the Predicted sliding window size, which may cause some problems. So in this section, with the **GRU encoder Decoder PR** model, we make an experiment on the evolution of loss function when different size are used for Past sliding window size and Predicted sliding window size.

Past sliding window size	Predicted sliding window size	MSE train $\times 10^{-2}$ avg over sum pitch and roll	MSE validation $\times 10^{-2}$ avg over sum pitch and roll	MSE test $\times 10^{-2}$ avg over sum pitch and roll
5	5	0.21583394	0.23295323	0.24630132
10	5	0.17880437	0.19861692	0.21303571
15	5	0.17493936	0.19289981	0.21668173

Table 6.8: Evolution of loss function: *Red cell:* worst result among the three; *Green cell:*, best result among the three

It can be seen from the table above that when the size of the Past sliding window is smaller than the size the Predicted sliding window, which means that the duration of the ship motion we want to predict is longer than the history we've already have, the result is worse than when we have a history longer than the future.

In this internship, a short history is used to predict a longer future, for lack of enough data, we can not have enough amount of long history (which decrease the number of groups of training data set). Another problem also occurs, when the sum of the size of Past sliding window and that of Predicted sliding window is too big (eg. >40), the training loss will be always zero, which may be a problem of data loader. Hoping that in the future work this problem can be solved.

6.6 Results

Model **CNN GRU encoder decoder images PR** and **CNN LSTM encoder attention decoder images PR*** are chosen for further analysis as the most suitable because of good convergence properties, it is balanced in prediction of pitch and roll and the loss function reaches one of its minimal values. And finally it's the **CNN GRU encoder decoder images PR** that is chosen for it's better performance. The Hyperband algorithm described in Section 5.5 was used to search hyper-parameter.

Hyperparameters	Distribution	[min, max]
Encoder latent vector	uniform	[500, 1500]
Decoder latent vector	uniform	[250, 500]
Learning rate	uniform	$[9 \cdot 10^{-6}, 2 \cdot 10^{-3}]$
Weight decay	uniform	$[9 \cdot 10^{-5}, 3 \cdot 10^{-3}]$

Table 6.9: Hyperparameters search space

After the search was completed, the following best configuration was found.

Hyperparameters	Best configuration
Encoder latent vector	879
Decoder latent vector	475
Learning rate	0.0002494405179826537
Weight decay	0.00100294079046556

Table 6.10: Best configuration for CNN GRU encoder decoder images PR model

Then, using the best configuration, the final test of the model was carried out to predict 30 seconds ahead. Results are presented below.

	At 15th second	At25^{th} second
Pitch MSE [denormalized]	8.926350768302964	7.811104251111423
Roll MSE [denormalized]	9.510666848697195	8.974565325207754

Table 6.11: Final Test result

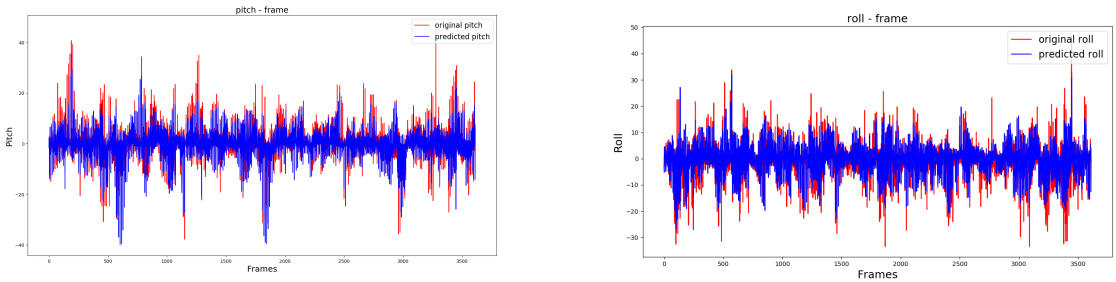


Figure 6.9: Original and predicted pitch (left) and roll (right) at 15th second using **CNN GRU encoder decoder images PR** model; *Red*: original labels, *Blue*: Predicted labels

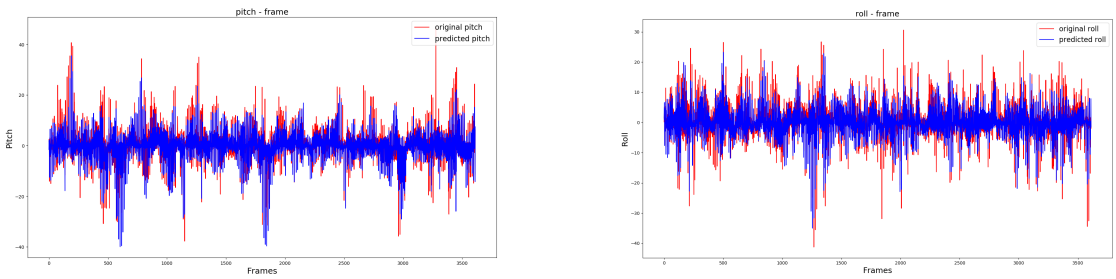


Figure 6.10: Original and predicted pitch (left) and roll (right) at 25th second using **CNN GRU encoder decoder images PR** model; *Red*: original labels, *Blue*: Predicted labels

Non-confidential report and publishable on the Internet

7 Conclusion

The internship focus on the design of deep learning models to predict the ship motion in a period of time. Based on the previous work of CNN and LSTM models with sequence length of size 1, sixteen deep neural network models were created and tested, where GRU, Attention mechanism and Transformer Model have been used. The sequence length of models are also changed from 1 to N. Such variety of models is caused by the complexity of the problem.

Empirical results show that GRU performs better in some certain small data set. For a simple model without the CNN part, it's the **GRU encoder decoder PR** that can make the prediction the most accurately. The results also proves that using additional information such as the current ship motion improve the pitch and roll prediction accuracy. The models with a CNN part perform much better than those without a CNN part.

The attention mechanism helps solving the problem of long term dependencies in the models of N sequence length, however, in those of 1 sequence length, the effect is not very obvious. The Transformer Model don't use directly a RNN neural network, the result is always the less satisfactory, which may be caused by the lack of enough data set.

The best results were achieved by a **CNN GRU encoder-decoder images PR** model and a **CNN LSTM encoder attention decoder images PR many** model. The best combination of parameters was found using Hyperband algorithm (learning rate, weight decay, encoder latent vector size and decoder latent vector size). In general, the models normally exhibit fluctuations, and sometimes skip large peaks. The problem is still not completely solved and can have many improvements. Even in the best version of the created model, problems such as overfitting, poor generalization, etc still exist. Besides, the models of Transformer is not made full use of in this internship, it should be changed more to adapt to our case. To achieve a better result, real data from the ship is also needed.

To sum up, after the creation and adjustment of various models, reasonable predictions have been accomplished with the proposed models, and many improvements can be made.

7.1 Improvements

Working on this project, number of aspects like models and configurations were investigated to obtain the best result. It is found that the use of Attention Mechansim improves the result in N-sequence-length models. In the future we should continue to work on Attention mechanism and Transformer Model. More changes should be made on Transformer Model to adapt it our case like in this article [23]. Hyper-parameters: learning rate, weight decay, past/future window size were tested but other hyperparameters could be tested like: batchsize, optimizer, etc.

Non-confidential report and publishable on the Internet

A number of improvements can be done to continue this project. After working on the problem I want to offer the following ideas that may be useful:

1) Generate better data sets. The best option would be using real data from different ships. If not available, have a try with Unreal Engine graphics generator. It's more difficult to use Unreal Engine but When using it, pay attention to creating the physics model of the ship and the buoyancy effect, which greatly affects the behavior of the ship and the sea.

   In this internship, a short history is used to predict a longer future, for lack of enough data, we can not have enough amount of long history (which decrease the number of groups of training data set). Besides the problem already mentioned, When the sum of the size of Past sliding window and that of Predicted sliding window is too big (eg. > 40), the training loss will be always zero, which may be a problem of data loader. Hoping that in the future work this problem can be solved.

2) Make full use of Transformer Model. Even though the transformer model created in this internship doesn't perform well, there are already a lot of successful forecasting examples with Transformer. The architecture of Transformer model should be changed to be adapted to our case, like the increase or decrease the number of the layers. The data set could also be changed for better utilization of Transformer model.

3) More parameters of the models of sequence length of size N should be tested to have a better performance. The Attention mechanism has a good effect on the performance of the N-sequence-length-models, so more attention should be paid to the improvement of the structure and the parameters.

   In this internship, Attention mechanism is used in the part of RNN to connect previous hidden stated with the previous one, it can also be used in the CNN part, which helps to improve the accuracy of extract the features of images, which deserves a try in the future work.

Non-confidential report and publishable on the Internet

References

[1] S. Wibowo and H. Deng, "Intelligent decision support for effectively evaluating and selecting ships under uncertainty in marine transportation," *Expert Systems with Applications*, vol. 39, no. 8, pp. p.6911–6920, 2012. URL: https://www.sciencedirect.com/science/article/abs/pii/S095741741200005X.

[2] G. Zhao, R. Xu, and C. Kwan, "Ship-motion prediction: Algorithms and simulation results," *IEEE International Conference on Acoustics, Speech*, vol. 5, p. 125, 2004. URL: https://www.researchgate.net/publication/4087942_Ship-motion_prediction_Algorithms_and_simulation_results.

[3] F. Cipollini, L. Oneto, A. Coraddu, and S. Savio, "Unsupervised deep learning for induction motor bearings monitoring," *Data-Enabled Discovery and Applications*, vol. 3, no. 1, 2019. URL: https://www.sciencedirect.com/science/article/pii/S1877050918322129.

[4] L. Oneto, I. Buselli, A. Lulli, R. Canepa, S. Petralli, and D. Anguita, "A dynamic, interpretable, and robust hybrid data analytics system for train movements in large-scale railway networks," *International Journal of Data Science and Analytics*, 2019. URL: https://link.springer.com/article/10.1007/s41060-018-00171-z.

[5] A. Zamani, D. Solomatine, A. Azimian, and A. Heemink, "Learning from data for wind-wave forecasting," *Ocean Engineering*, vol. 35, no. 10, pp. p.953–962, 2008. URL: https://www.sciencedirect.com/science/article/abs/pii/S0029801808000632.

[6] S. Oehmcke, O. Zielinski, and O. Kramer, "Input quality aware convolutional lstm networks for virtual marine sensors," *Neurocomputing*, vol. 275, pp. 2603–2615, 2018. URL: https://www.sciencedirect.com/science/article/abs/pii/S0925231217317861.

[7] Z. Zhou, "Predicting ship's motion based on machine learning," 2018. URL: https://drive.google.com/file/d/1A0Jgqq1ElIiHYhDEImBzrKtDDpGBwnfp/view.

[8] N.-M. Kaminskyi, "Deep learning for predicting ship motion from images," 2019. URL: https://drive.google.com/file/d/12S02yDJ_CpNKnAeWrjgrIN4fMvdv-ySk/view?usp=sharing.

[9] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," *Advances in Neural Information Processing Systems*, vol. 3, 06 2014. arXiv: 1406.6247.

[10] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," *Computer Science*, pp. 2048–2057, 2015. arXiv: 1502.03044.

[11] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," 2017. arXiv: 1704.02971.

[12] F. Tharindu, D. Simon, S. Sridha, and F. Clinton, "Soft + hardwired attention: An lstm framework for human trajectory prediction and abnormal event detection.," *Neural networks: the official journal of the International Neural Network Society*, 2018. arXiv: 1702.05552.

[13] F. Tharindu, D. Simon, S. Sridha, and F. Clinton, "Tracking by prediction: A deep generative model for mutli-person localisation and tracking," *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018. arXiv: 1803.03347.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. arXiv: 1706.03762.

[15] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in neural information processing systems*, 2014. URL: https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

[16] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," 2019. URL: http://www.d2l.ai.

[17] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *ArXiv e-prints*, 11 2015. arXiv: 1511.08458.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.

[19] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," 2013. arXiv: 1311.2901.

[20] M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light gated recurrent units for speech recognition," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, p. 92–102, Apr 2018. DOI: 10.1109/tetci.2017.2762739.

[21] N. Gruber and A. Jockisch, "Are gru cells more specific and lstm cells more sensitive in motive classification of text?," *Frontiers in Artificial Intelligence*, vol. 3, p. 40, 2020. DOI: 10.3389/frai.2020.0004.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. arXiv: 1412.6980.

[23] N. Wu, B. Green, X. Ben, and S. O'Banion, "Deep transformer models for time series forecasting: The influenza prevalence case," 2020. arXiv: 2001.08317.

Non-confidential report and publishable on the Internet

List of Figures

1.1 Internship mission 13
3.1 Illustration of rotational and translational motion from wikipedia 16
3.2 Illustration of images generated by Blender 17
3.3 Illustration of ocean models generated by Unreal Engine 18
4.1 Illustration of LeNet 21
4.2 The repeating module in an LSTM contains four interacting layers: 1) sigmoid layer (f_t); 2) sigmoid layer (i_t); 3) tanh layer ($\tilde{C}_t$); 4) sigmoid layer (o_t). 21
4.3 LSTM notions 22
4.4 Architecture of GRU module 24
4.5 Illustration of the architecture of a transformer model 25
4.6 Illustration of the encoder of models with sequence length of size 1 . . . 27
4.7 Illustration of the encoder of models with sequence length of size N . . . 27
4.8 Illustration of the decoder of the models with sequence length of size 1 . 28
4.9 Illustration of the decoder of the models with sequence length of size N . 28
5.1 Illustration of data iteration step by step for models with sequence length of size 1 39
5.2 Illustration of data iteration step by step for models with sequence length of size N 39
6.1 Loss function for [GRU encoder decoder PR] model 42
6.2 Loss function for [CNN LSTM encoder decoder images PR] model 43
6.3 Loss function for [LSTM encoder decoder images PR*] model 45
6.4 Loss function on the **training** and the **validation** data for all models . . 46
6.5 Loss function on the **training** data with **test** results the best models . . 47
6.6 Original and predicted pitch (left) and roll (right) at 10th second using **LSTM encoder decoder PR** model; *Red*: original labels, *Blue*: Predicted data. 48
6.7 Original and predicted pitch (left) and roll (right) at 10th second using **CNN GRU encoder decoder images PR roll** model; *Red*: original labels, *Blue*: Predicted labels 48
6.8 Original and predicted pitch (left) and roll (right) at 10th second using **CNN LSTM encoder attention decoder images PR*** model; *Red*: original labels, *Blue*: Predicted labels 48
6.9 Original and predicted pitch (left) and roll (right) at 15th second using **CNN GRU encoder decoder images PR** model; *Red*: original labels, *Blue*: Predicted labels 50
6.10 Original and predicted pitch (left) and roll (right) at 25th second using **CNN GRU encoder decoder images PR** model; *Red*: original labels, *Blue*: Predicted labels 50
A.1 Architecture of [LSTM/GRU encoder decoder PR] model 59
A.2 Architecture of [CNN LSTM/GRU encoder decoder images PR] model . 59

A.3 Architecture of [LSTM/GRU encoder attention decoder PR] model . . . 60
A.4 Architecture of the Attention Decoder 60
A.5 Architecture of the Attention layer 61
A.6 Architecture of [CNN LSTM/GRU encoder attention decoder images PR] model 61
A.7 Architecture of [LSTM/GRU encoder decoder PR*] model 62
A.8 Architecture of LSTM/GRU decoder* 62
A.9 Architecture of [CNN LSTM/GRU encoder decoder images PR*] model 63
A.10 Architecture of [LSTM/GRU encoder attention decoder PR*] model . . 63
A.11 Architecture of Attention decoder* 64
A.12 Architecture of [CNN LSTM/GRU encoder attention decoder images PR*] model 64

Non-confidential report and publishable on the Internet

List of Tables

3.1 Camera parameters in data generation 17
3.2 Image resolution 18
4.1 LSTM encoder decoder PR model architecture in previous work 23
4.2 CNN GRU encoder decoder PR model architecture 23
4.3 Models nomenclature for each module component 26
4.4 Proposed Models 26
4.5 GRU encoder decoder PR model architecture 29
4.6 LSTM encoder attention decoder PR model architecture 29
4.7 Attention layer architeture 30
4.8 GRU encoder attention decoder PR model architecture 30
4.9 CNN GRU encoder decoder PR model architecture 31
4.10 CNN LSTM encoder attention decoder images PR model 31
4.11 CNN LSTM encoder decoder PR* architecture 32
4.12 CNN LSTM encoder GRU attention decoder images PR model 32
4.13 LSTM encoder decoder PR model architecture version 1 33
4.14 LSTM encoder decoder PR* architecture 33
4.15 GRU encoder decoder PR* architecture 34
4.16 LSTM encoder attention decoder PR* architecture 34
4.17 LSTM encoder GRU attention decoder PR* architecture 35
4.18 CNN LSTM encoder decoder images PR* 35
4.19 CNN LSTM encoder decoder PR* architecture 36
4.20 CNN LSTM encoder GRU decoder images PR* architecture 36
4.21 TransformerModel PR model architecture 37
5.1 Models nomenclature for each module component 38
6.1 Basic hyper-parameters for models 41
6.2 Results for models with only one time step but without a CNN part. *Red line* - our baseline [LSTM encoder decoder PR] model, worst result; *Green line* - the best result at the moment 42
6.3 Results for models with sequence length of size 1 and with a CNN part. *Red line* - our baseline [CNN LSTM encoder decoder images PR] model, *Green line* [CNN GRU encoder decoder images PR] model, best model created 43
6.4 Results for models with many time steps. *Red line* - [LSTM encoder GRU attention decoder PR*] model, worst model. *Green line* - [CNN LSTM encoder attention decoder images PR] model, best model. 44
6.5 Results for models without direct use of RNN. 45
6.6 Comparison of the best models in different type. *Red line* - [LSTM encoder decoder PR] model, our baseline model; *Strong green line* - [CNN GRU encoder attention decoder images PR] model, the best model at this moment; *Light green line* - [CNN LSTM encoder attention decoder images PR*] model, second best model at this moment. 46

6.7 Comparison of the best models in different type. *Red line* - [LSTM encoder decoder PR] model, our baseline model; *Strong green line* - [CNN GRU encoder attention decoder images PR] model, the best model at this moment; *Light green line* - [CNN LSTM encoder attention decoder images PR*] model, second best modelat this moment. 47
6.8 Evolution of loss function: *Red cell:* worst result among the three; *Green cell:*, best result among the three 49
6.9 Hyperparameters search space 49
6.10 Best configuration for CNN GRU encoder decoder images PR model . . . 50
6.11 Final Test result 50

Non-confidential report and publishable on the Internet

Appendix A Architectures of models

A.1 Architectures of models with sequence length of size 1

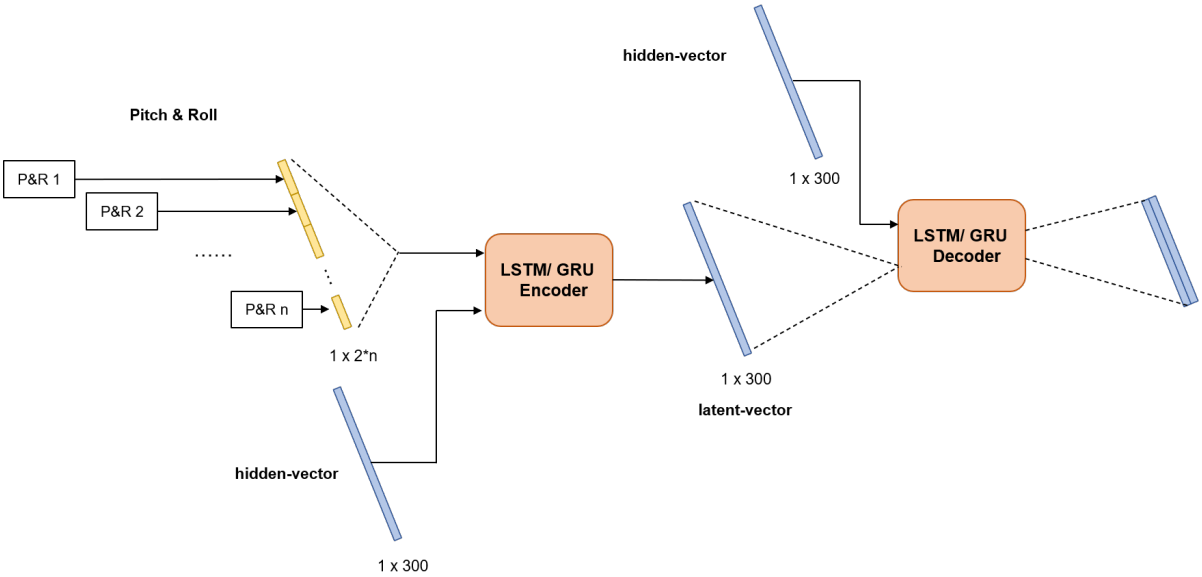


Figure A.1: Architecture of [LSTM/GRU encoder decoder PR] model

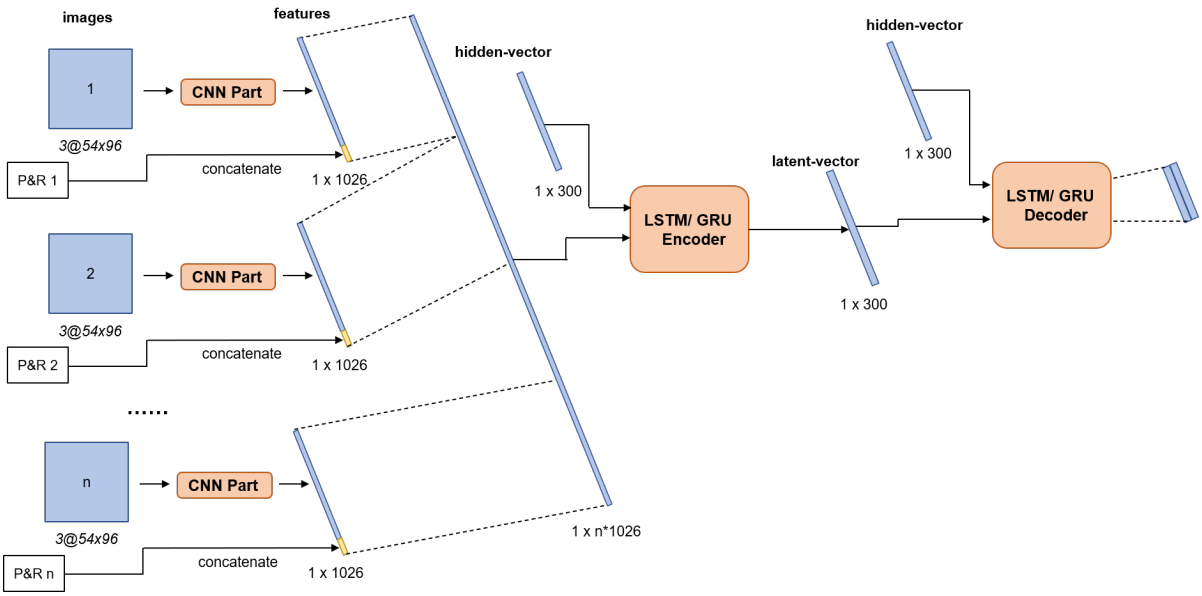


Figure A.2: Architecture of [CNN LSTM/GRU encoder decoder images PR] model

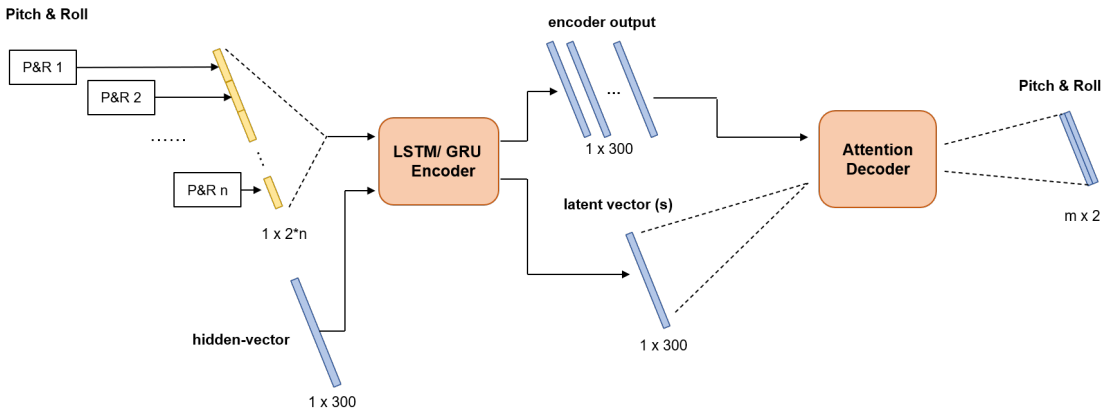


Figure A.3: Architecture of [LSTM/GRU encoder attention decoder PR] model

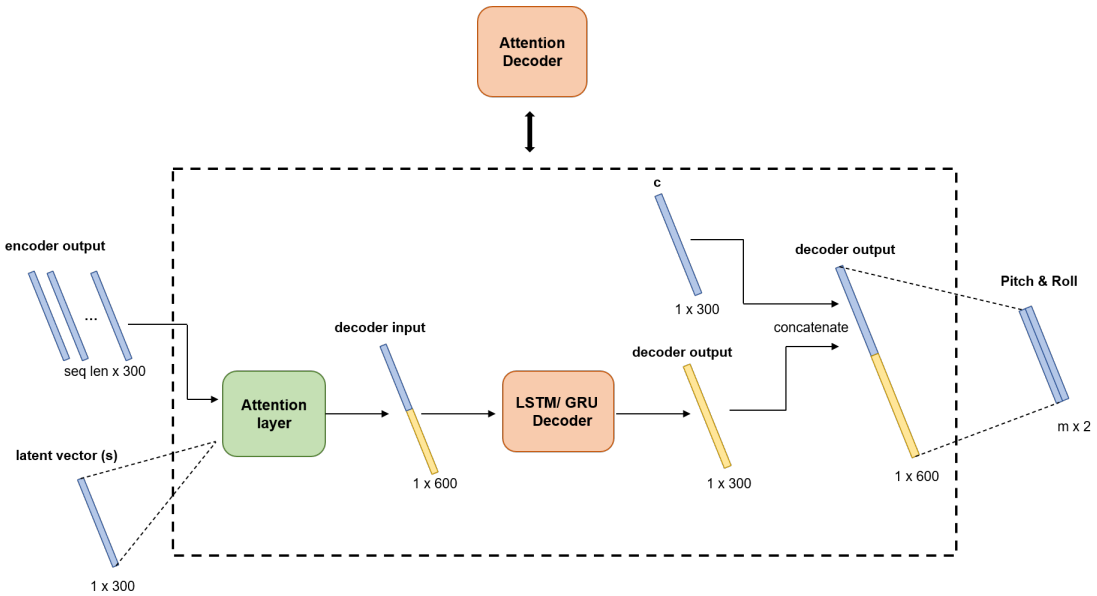


Figure A.4: Architecture of the Attention Decoder

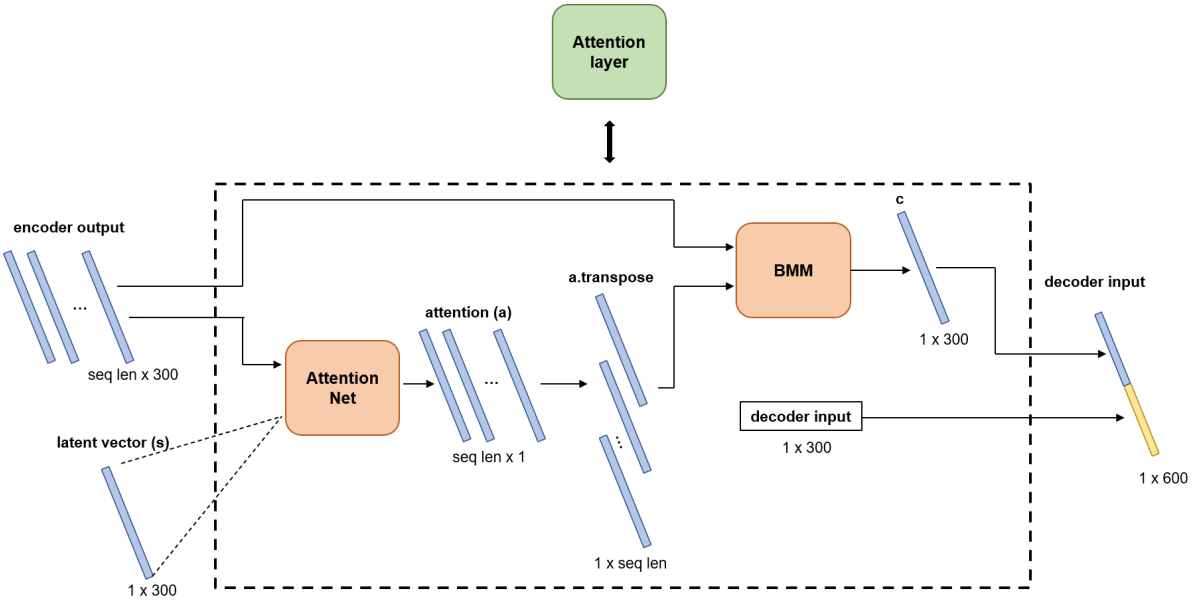


Figure A.5: Architecture of the Attention layer

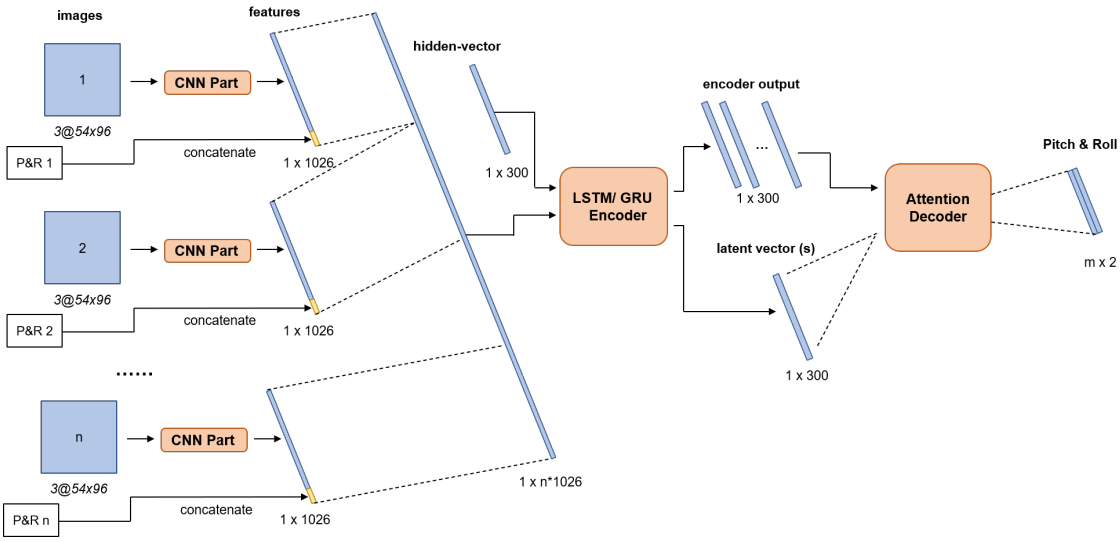


Figure A.6: Architecture of [CNN LSTM/GRU encoder attention decoder images PR] model

A.2 Architectures of models with sequence length of size N

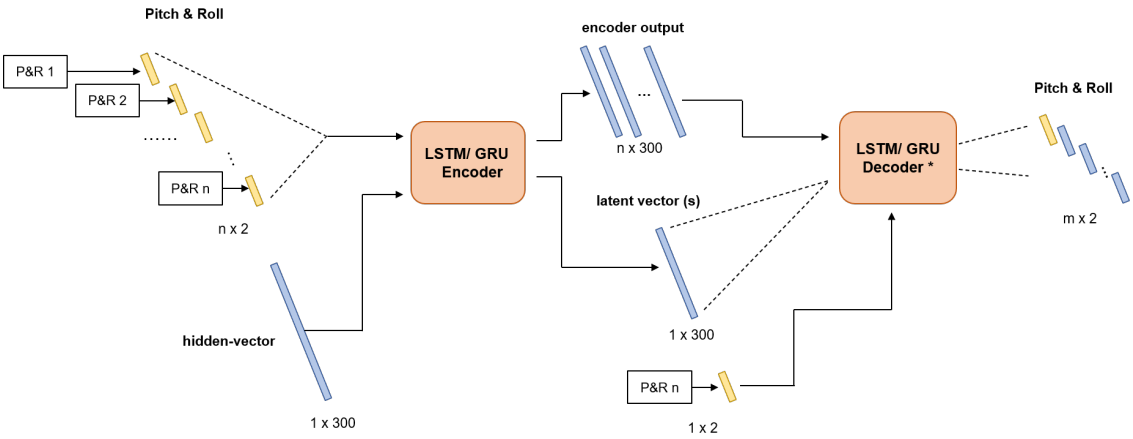


Figure A.7: Architecture of [LSTM/GRU encoder decoder PR*] model

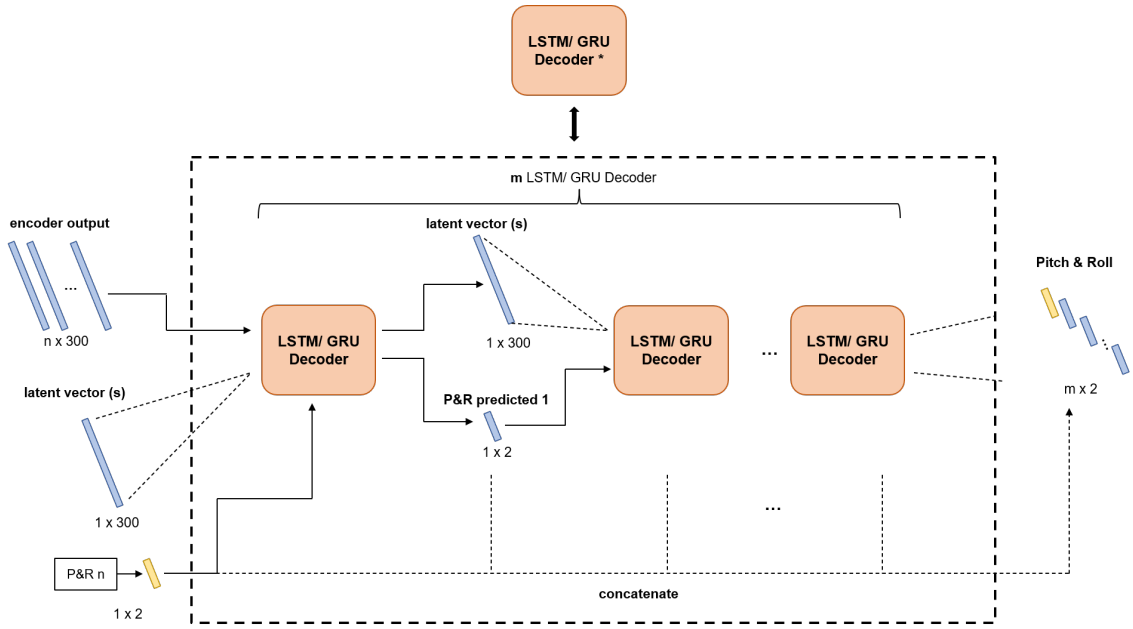


Figure A.8: Architecture of LSTM/GRU decoder*

Non-confidential report and publishable on the Internet

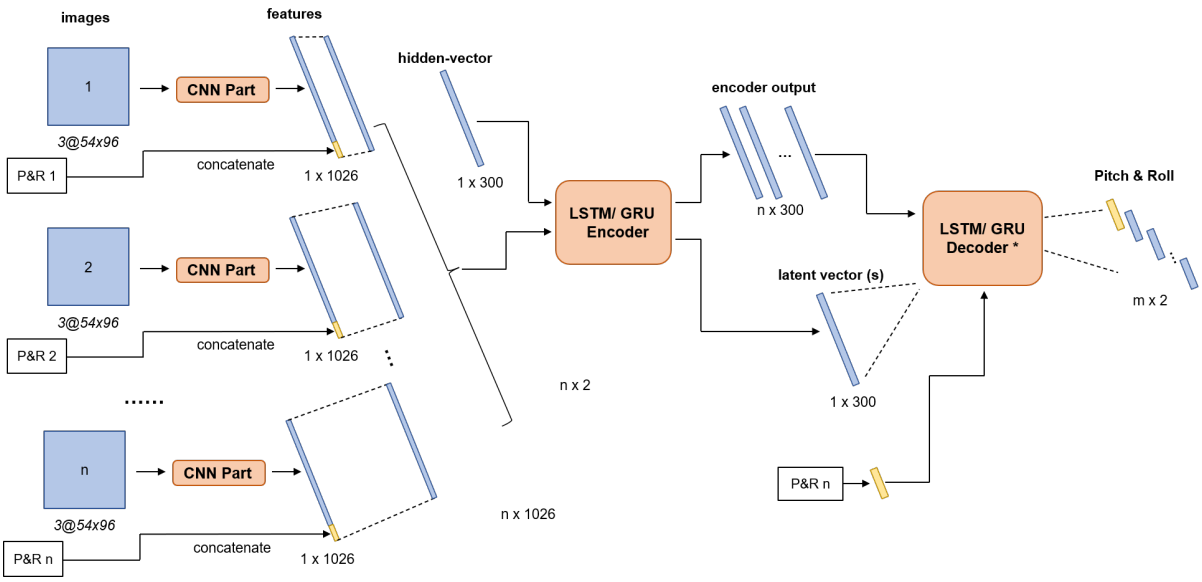


Figure A.9: Architecture of [CNN LSTM/GRU encoder decoder images PR*] model

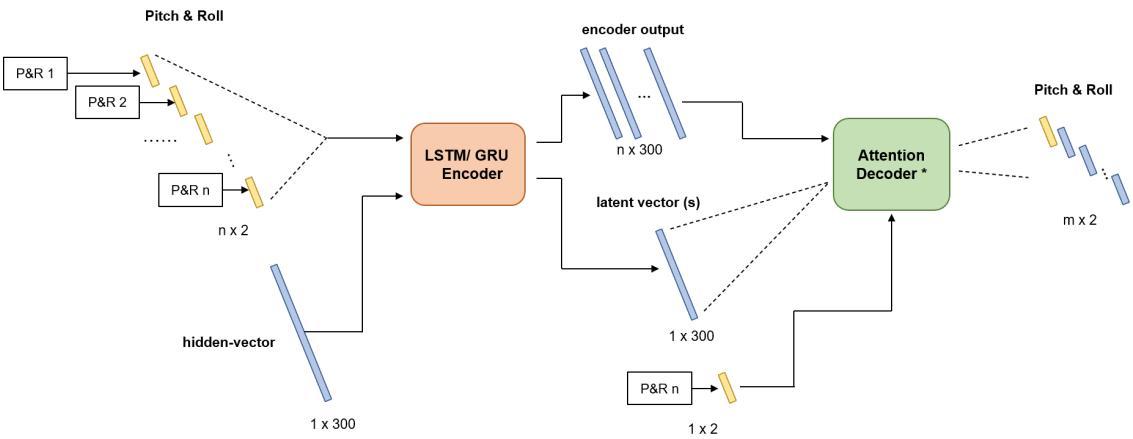


Figure A.10: Architecture of [LSTM/GRU encoder attention decoder PR*] model

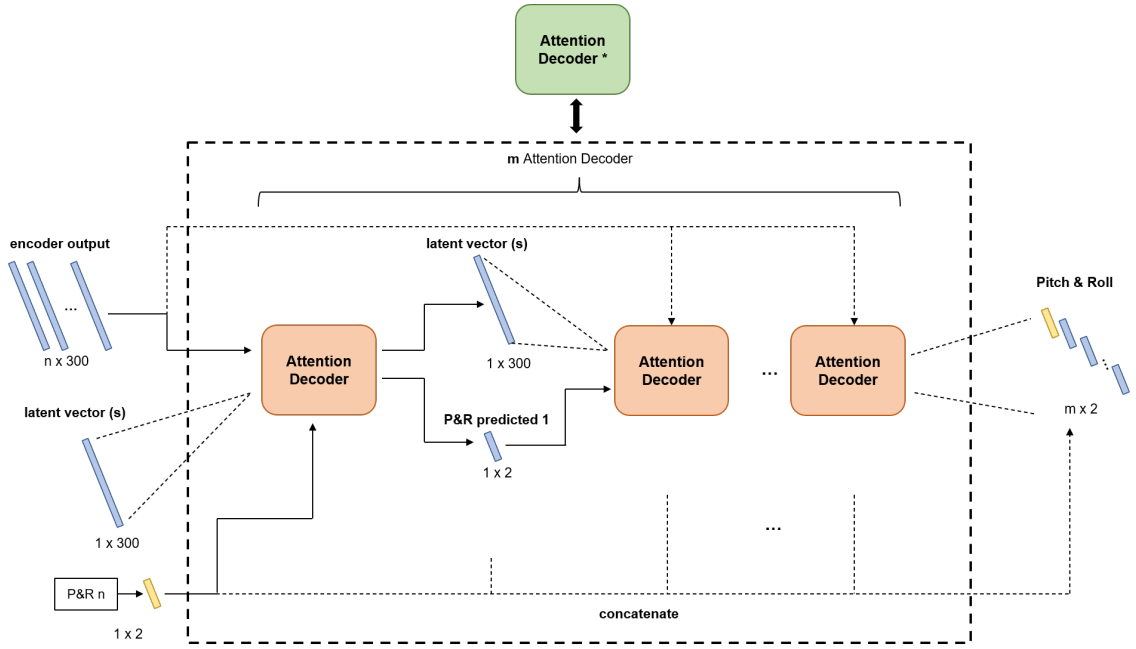


Figure A.11: Architecture of Attention decoder*

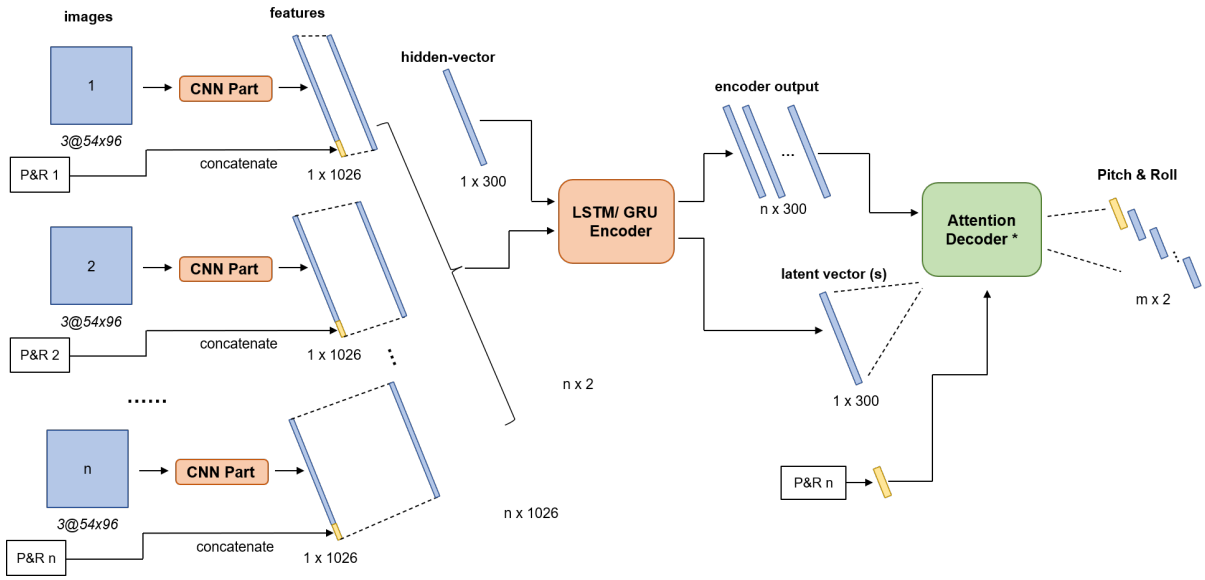


Figure A.12: Architecture of [CNN LSTM/GRU encoder attention decoder images PR*] model