Ising model

Recognizing Temparature labels by CNN model

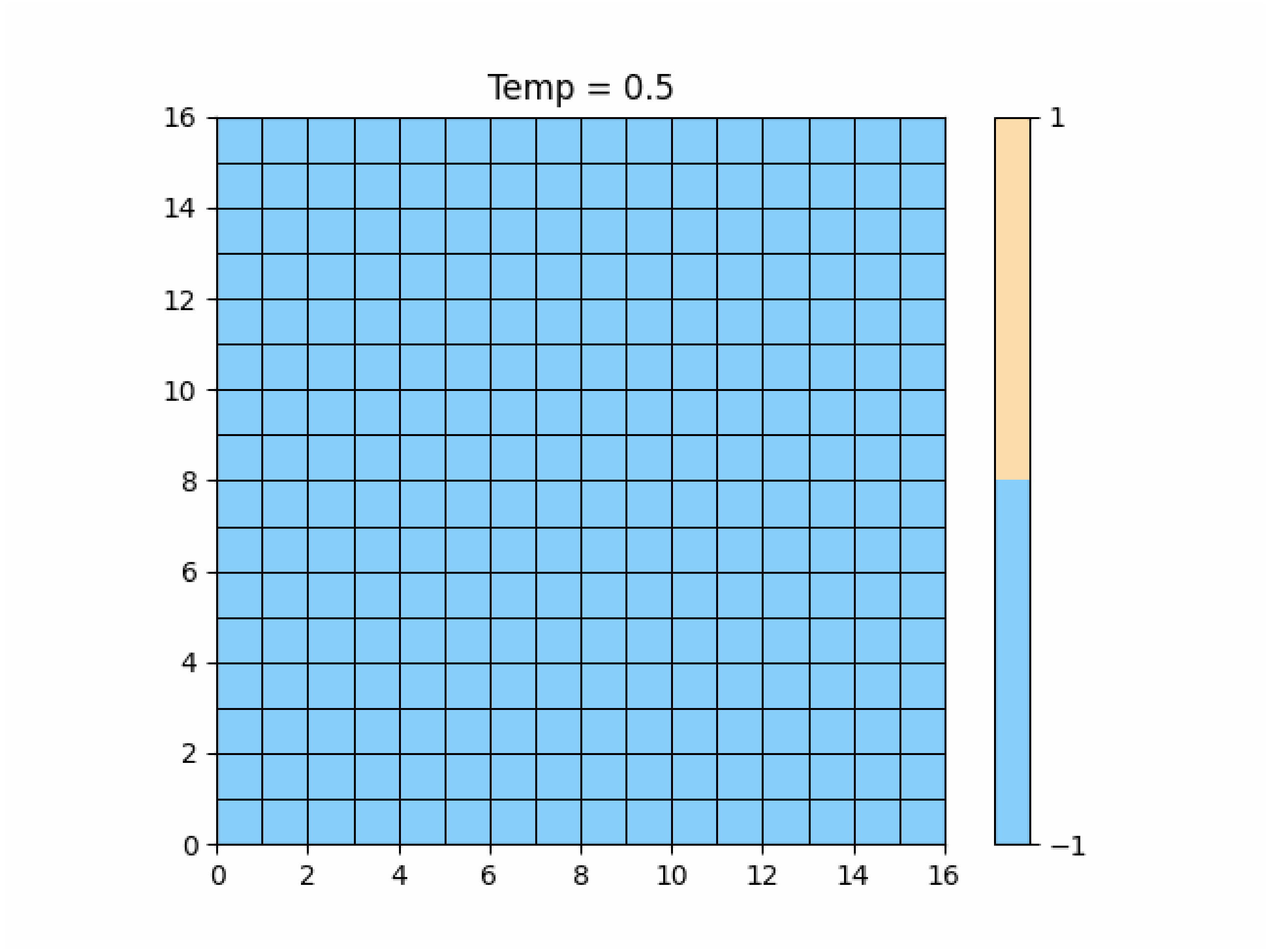


Figure 1: spinning direction of lattice points in size 16x16 changes with the time

Submitted by:

LIN XIAO-DAO

Contents

1 Introduction

1.1 Motivation for the topic

All the topics given by the teacher in class were very interesting, which actually made me quite confused when choosing a topic. However, considering that I am familiar with the image recognition aspect of machine learning and am eager to apply it to real physical problems (although I have also worked on other topics), I chose this topic.

I am also very happy that the teacher assigned me the topic of identifying Ising model temperature labels. Compared to when I took the course in my senior year, I now have a deeper understanding and grasp of the Ising model, so I am more confident in completing this topic on my own. And as I worked on it, I found it very enjoyable.

1.2 Ising model

The Ising model is a mathematical physics model used to study phase transitions and symmetry breaking in magnetic materials. It was proposed by German physicist Ernst Ising in 1925. The model consists of a series of spins arranged on a lattice, where each spin can take one of two values (usually represented as +1 or -1), corresponding to two opposite magnetic states.

The core of the Ising model is the interaction between spins: neighboring spins tend to align or anti-align depending on the interaction energy. As the temperature decreases, the system tends to reach a lower energy, ordered state (i.e., most spins align); as the temperature increases, thermal fluctuations cause the system to be in a disordered state (spins are randomly distributed).

The Ising model is crucial in understanding phase transitions, which are changes between different states of matter. At a critical temperature, the system undergoes a phase transition from a disordered (high-temperature) phase to an ordered (low-temperature) phase. This transition is a classic example of symmetry breaking: at high temperatures, the system is symmetric because the spins are randomly oriented and no particular direction is favored. As the temperature lowers and the system becomes ordered, this symmetry is broken because the spins align in a specific direction, indicating a preferred state.

This model is significant in statistical physics because it captures the behavior of phase transitions and symmetry breaking, and has broad applications in the study of critical phenomena and phase transition theory.

1.3 Convolution Neural Networks (CNN)

A Convolutional Neural Network (CNN) is a type of deep learning model that is particularly well-suited for processing image data. It has widespread applications in image recognition and computer vision.

1. Convolutional Layer: The convolutional layer is the fundamental building block of CNNs. It uses convolutional kernels (also called filters) that slide over the image to extract local features, such as edges and corners. Each kernel extracts different features and generates a feature map.
2. Pooling Layer: The pooling layer typically follows the convolutional layer and is used to reduce the dimensions of the feature maps, thereby reducing the computational load and the risk of overfitting. The most common pooling operation is max pooling, which takes the maximum value in a local region.
3. Fully Connected Layer: In the final stages of a CNN, fully connected layers integrate the extracted features and use a Softmax function to produce the final classification results.

The workflow of a CNN is roughly as follows:

1. Image data is input into the model, which passes through multiple convolutional and pooling layers to extract features.
2. The extracted features are integrated and classified through fully connected layers.
3. The final output is produced, such as the category of the object in the image.

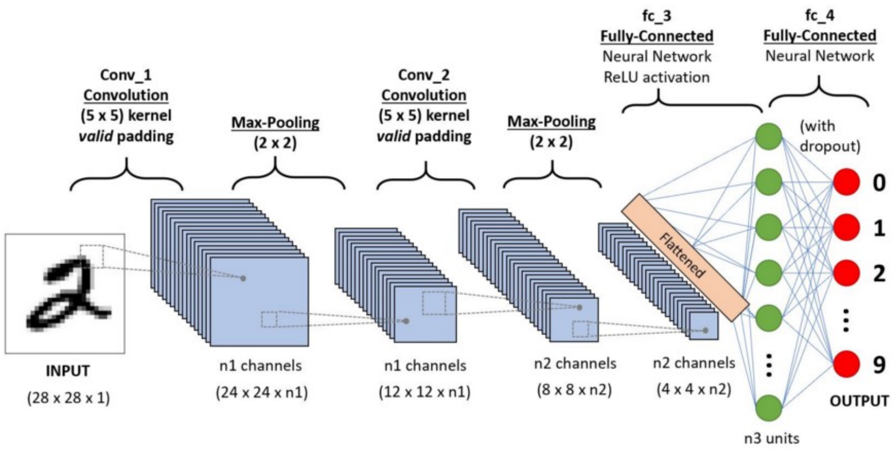


Figure 1.1: An example of the workflow of CNN model [1]

2 Generate data from Ising model

Regarding Metropolis-Hastings algorithm, I have already explained and described it in class and in the assignments. I only added functionality to return the lattice matrix at each Monte Carlo step (along with extensive testing of Numba). Therefore, this part of the program will not be further explained.

Additionally, concerning whether to use different lattice sizes as input data for the CNN model, as long as the lattice is not too small to lack sufficient information, considering the relationship between lattice size and the model, the difference might only be in the size of the model parameters required for effective training. (Of course, if we could provide the same model with lattices of different input sizes, that would be ideal, but for convenience, we won't test this here.) Therefore, we will use a single lattice size as the input data.

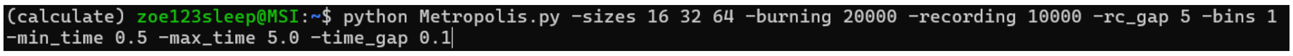
Next, we will focus on selecting appropriate parameters for Metropolis-Hastings algorithm and the results of the final data generation.

2.1 Parameters of Metropolis-Hastings algorithm

In my Metropolis-Hastings algorithm code (filename Metropolis.py), several important parameters are included(the variable names will be attached next to the description):

- the lattice sizes(sizes)
- the minimum and maximum temperature range(min_time, max_time)
- the temperature interval (time_gap)
- the Monte Carlo steps not recorded (burning)
- the Monte Carlo steps recorded (recording)
- the recording interval (rcgap)
- the number of independent experiments to repeat at each temperature (bins)

Here, the argparse package is used for parameter management, allowing them to be generated directly from the command line. I first tested the parameters bellow to check the stability of the result:



```
(calculate) zoe123sleep@MSI:~$ python Metropolis.py -sizes 16 32 64 -burning 20000 -recording 10000 -rc_gap 5 -bins 1
-min_time 0.5 -max_time 5.0 -time_gap 0.1
```

Figure 2.1: The first testing parameters

To check whether the Metropolis-Hastings algorithm converges well enough, we use four physical quantities—lattice energy E, heat capacity C, magnetization M, and magnetic susceptibility X —to observe the results, as shown in Figure 2-2.

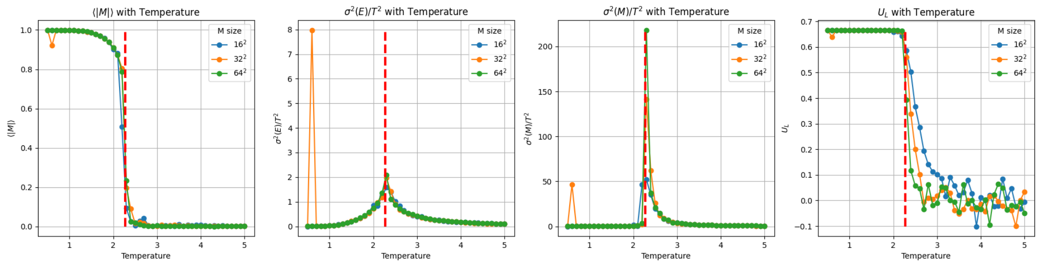
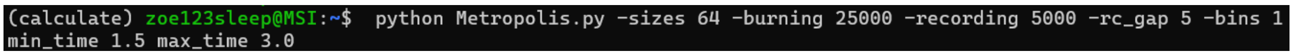


Figure 2.2: Check the stability of the result

The total duration of this experiment was approximately 26 minutes, with the lattice sizes consuming 1, 5, and 20 minutes respectively from smallest to largest. We can observe that, apart from the lattice of size 16 showing anomalous spikes at low temperatures and the lattice of size 32 being somewhat unstable, overall, the three lattice sizes performed relatively stably at different temperatures. This is good news, indicating that the algorithm is able to converge quite well under the current parameter settings, though some statistical errors still exist.

Considering the results this time, in order to address statistical errors, achieve better convergence of the algorithm, and take into account the computer's performance, it has been decided to adjust the burning variable to 25,000 and the recording variable to 5,000. Additionally, to ensure convergence to multiple scenarios at the same temperature, the bins variable will be adjusted to 5. Finally, considering that the magnetization M shows almost no change when the temperature is below 0.5 and above 3.0, it has been decided to restrict the temperature range to $[1.5, 3.0]$. Other variables will use the settings from the previous experiment.



```
(calculate) zoe123sleep@MSI:~$  python Metropolis.py -sizes 64 -burning 25000 -recording 5000 -rc_gap 5 -bins 1
min_time 1.5 max_time 3.0
```

Figure 2.3: The determined parameters

2.2 Final Data information

- Size : $(80 * 1000, 64, 64)$
- Consuming Time : 30 mins
- Train data : use the first 50k datasets
- Test data : use the full datasets

3 CNN model and training

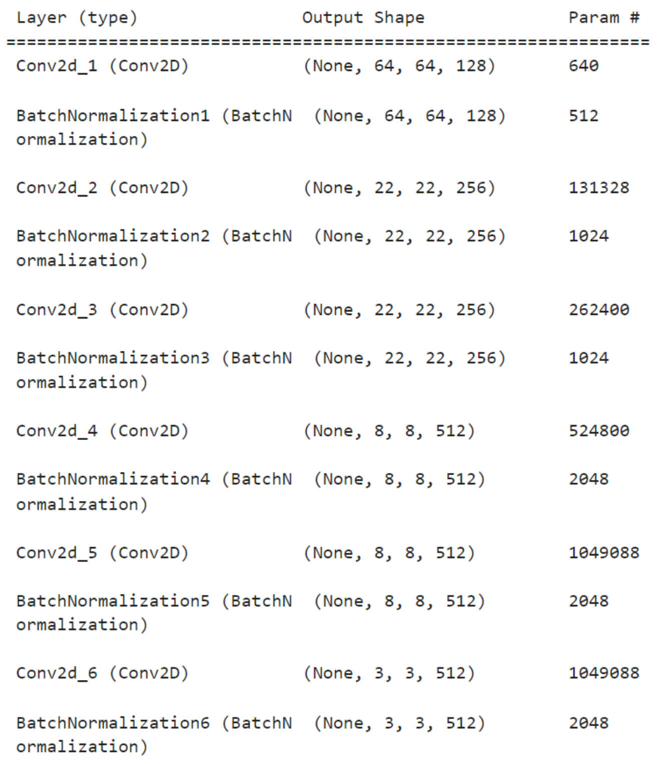
3.1 Design policy

In the model design, I aim to create a simple model that doesn't require too many parameters (i.e., not too deep, with possibly fewer than 10 million parameters) and can achieve a prediction accuracy of over 70%. This would make the explanation clearer and maximize the benefits (though I later realized I made a mistake).

Given that my input data size is a 64x64 matrix, to avoid making the model too deep, I chose a stride of 3 and set the filter sizes to vary in powers of 2. This allows the input data to rapidly shrink while capturing many features. Batch normalization is applied between convolutional layers to ensure that different features can be used as reference features on the same scale.(See details in Figure 3.1)

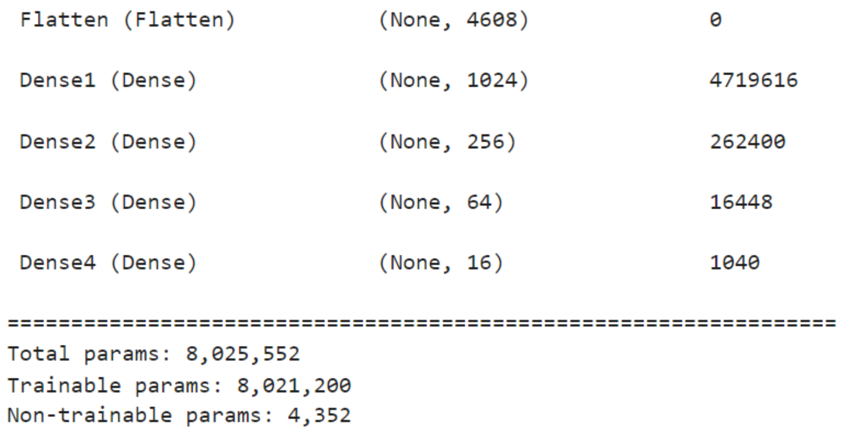
Next, I flattened the input data and designed fully-connected layers that shrink by powers of 2 with increasing depth. The final layer is set to 16 neurons, corresponding to our 16 temperature labels.(See details in Figure 3.2)

Throughout this process, the activation function used is the ReLU function. Many reports in the field of artificial intelligence use the ReLU function, as it enables the model to converge more easily compared to other activation functions.



```
 Layer (type)                Output Shape              Param #
=================================================================
 Conv2d_1 (Conv2D)           (None, 64, 64, 128)       640

 BatchNormalization1 (BatchN  (None, 64, 64, 128)      512
 ormalization)

 Conv2d_2 (Conv2D)           (None, 22, 22, 256)       131328

 BatchNormalization2 (BatchN  (None, 22, 22, 256)      1024
 ormalization)

 Conv2d_3 (Conv2D)           (None, 22, 22, 256)       262400

 BatchNormalization3 (BatchN  (None, 22, 22, 256)      1024
 ormalization)

 Conv2d_4 (Conv2D)           (None, 8, 8, 512)         524800

 BatchNormalization4 (BatchN  (None, 8, 8, 512)        2048
 ormalization)

 Conv2d_5 (Conv2D)           (None, 8, 8, 512)         1049088

 BatchNormalization5 (BatchN  (None, 8, 8, 512)        2048
 ormalization)

 Conv2d_6 (Conv2D)           (None, 3, 3, 512)         1049088

 BatchNormalization6 (BatchN  (None, 3, 3, 512)        2048
 ormalization)
```

Figure 3.1: Convolution part of CNN



```
 Flatten (Flatten)           (None, 4608)              0

 Dense1 (Dense)              (None, 1024)              4719616

 Dense2 (Dense)              (None, 256)               262400

 Dense3 (Dense)              (None, 64)                16448

 Dense4 (Dense)              (None, 16)                1040

=================================================================
Total params: 8,025,552
Trainable params: 8,021,200
Non-trainable params: 4,352
```

Figure 3.2: fully-connected layers of CNN and detailed info

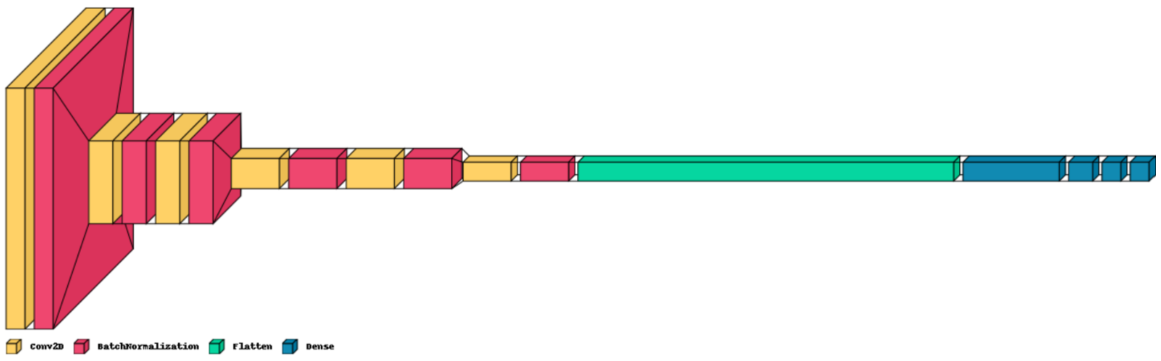


Figure 3.3: overview of the model

3.2 Training Details

- Epoch : 30
- Loss Function : SparseCategoricalCrossentropy
- Optimizer : Adam (learning rate = 0.001)
- Metric : Accuracy

4 Result

4.1 Loss curve and accuracy evaluation

Before examining the accuracy, let's first look at the loss curves for the training and testing data at each epoch.

Figure 4.1 shows that the model's loss decreases and tends to converge as the number of training epochs increases, indicating that the model design is fundamentally correct. However, we can also observe that the loss curve for the test data floats above the loss curve for the training data, except at the eighth epoch where they almost overlap. This suggests that the model is slightly overfitting. Therefore, the next improvement will focus on regularizing the fully-connected layers using L2 regularization(Figure 3.1), in hopes of resolving this issue.

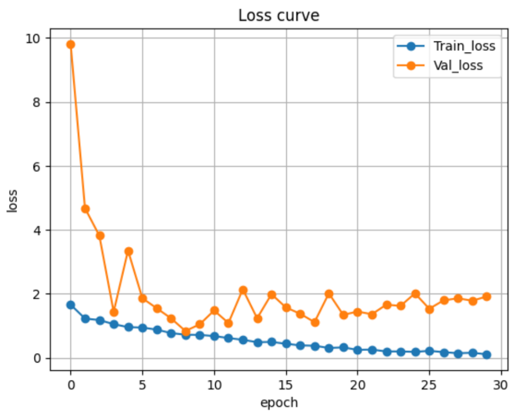


Figure 4.1: The loss curve of train and validation data

Due to the poor performance of the loss curve on the test data, we can also expect the accuracy to be low. As seen in Figure 4.2, although the training data can achieve almost 95% accuracy, the test data only reaches around 60%.

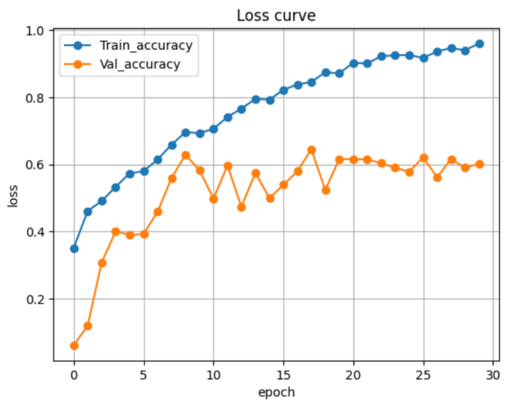


Figure 4.2: The accuracy of train and validation data

4.2 Improvement by Regularization

From Figure 4.2, we can see that after applying regularization to the fully-connected layers, the loss curve for the validation data is indeed closer to the loss curve for the training data! This is very exciting because it suggests that we might achieve results as good as the accuracy on the training data.

However, unfortunately, as seen in Figure 4.3, we find that the accuracy for the validation data is still struggling around 60%.

```
model2.add(Dense(name="Dense1", units=1024,
                 activation = 'relu', kernel_regularizer=keras.regularizers.l2(l=0.1)))
model2.add(Dense(name="Dense2", units=256,
                 activation = 'relu', kernel_regularizer=keras.regularizers.l2(l=0.1)))
model2.add(Dense(name="Dense3", units=64,
                 activation = 'relu', kernel_regularizer=keras.regularizers.l2(l=0.1)))
```

Figure 4.3: Regularization to fully-connected layers

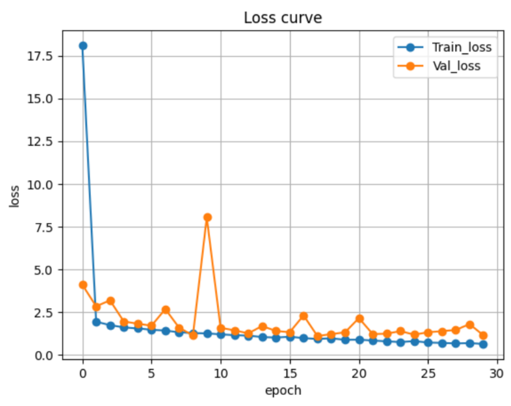


Figure 4.4: The loss curve of train and validation data

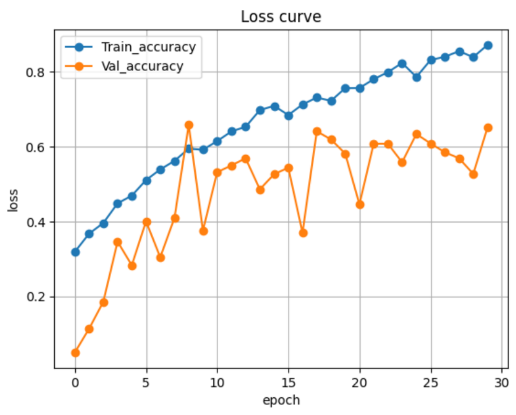


Figure 4.5: The accuracy of train and validation data

4.3 Accuracy for Prediction with different Temparature label

Finally, even though the previous model training results were not ideal, we can still explore the model's prediction accuracy for different temperature labels. I expect that we will see the most pronounced performance when the temperature is near low temperatures, high temperatures, or close to the critical temperature T_c point where changes are significant. In these cases, the model should be better able to recognize the temperature labels.

As shown in Figure 4.6, the model's prediction accuracy can reach over 80% when near the critical temperature T_c and the boundaries of our set temperature labels! This aligns with my expectations.

From this,we can also see that **if we could increase the amount of data in these transitional phases**, we might be able to improve our prediction accuracy. Of course, this is just a hypothesis.

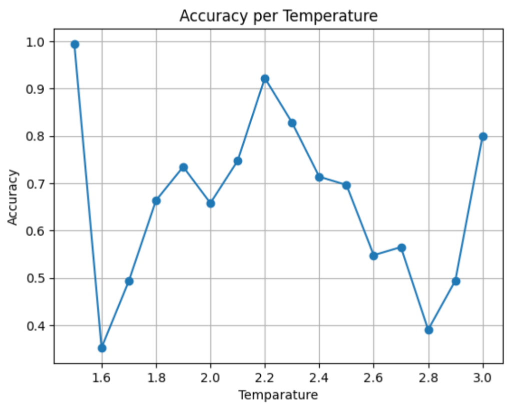


Figure 4.6: The accuracy of train and validation data

Summary

Due to having to handle research, moving, and part-time work in the days between presenting the report and submitting it, and because I wanted to learn LaTeX, I was unable to make more improvements. This is very disheartening (but fortunately, I did complete learning LaTeX).

I actually know there are many ways to improve. For example, I could have trained for a longer period. I initially chose 30 epochs because too much time was spent handling the data size on the hardware, leaving me with limited opportunities to experiment. I truly feel it's a pity because I know I could have done better.

But no matter what, I am happy that I was able to complete a comprehensive report. I am also very grateful for the help from the professor this semester, which gave me the opportunity to assist my classmates and receive more guidance from the professor. Sincerely, thank you, professor.

References

[1] J. C. Martinez, “Workflow of cnn.” Available at `https://blog.wordvice.com.tw/how-to-cite-images-apa-mla-chicago-vancouver/`.