CS230

Images As Thermometer: Temperature Prediction using Sequential Images (Computer Vision)

Yuwei Wu
Stanford University
yuweiwu@stanford.edu

Zilu Wang
Stanford University
zilu@stanford.edu

Hanxiao Zhao
Stanford University
hxzhao@stanford.edu

1 Introduction

Visual characteristics in images can give human immediate and intuitive perceptions – the bleak forest or white snow correlates to a cold winter; the blooming flowers or bright sunshine indicates a lush spring... However, it is extremely difficult for us to tell the subtle attributes within the image such as temperature in the surrounding environment. In this paper, we aim to use CNN to predict the ambient temperature given images of outdoor scenes. In addition, we want to combine CNN and LSTM as a multi-step approach in order to predict the temperature from a time series point of view. By feeding the pictures from the past few days and the picture for the subsequent day in the algorithm, we aspire to predict that subsequent day's temperature more accurately.

2 Related Work

Previous work in this field employed various deep learning models to explore the correlation between the image scene and the temperature[1], to recognize different types of daytime[2], to classify different types of weather[3], and even to distinguish sunrise and sunset[4]. Our work has been built upon their foundations by diving into the image compositions and trying to optimize the temperature prediction from images by using CNN. Beyond the previous established work, we innovated by implementing the LSTM algorithm based on our constructed CNN model to further improve the prediction accuracy.

3 Dataset

The Skyfinder dataset[5] consists of both images and relevant information such as temperature, meteorological indicators, and geographical locations. It consists of roughly 90,000 labeled images captured by 53 outdoor cameras under different weather and light conditions. This dataset also provides the associated weather information for each image, including the temperature data in Celsius degrees that we use in our model as labels.

In the pre-processing step, we eliminated all images with invalid temperature. In addition, during our first several trials in the model training process, we found our prediction results deviated a lot from the true temperatures. After some detailed error analysis, we found that the temperature prediction was extremely inaccurate for images taken during the night. Pictures taken at night were dimly and blurry, which provide little visual information, even for human inspection. Hence, we decided to focus on temperature prediction during the daytime by filtering for images that are taken between 10:00AM and 5:00PM. We also included the geographical location (longitude, latitude) and time associated with each picture into our CNN model to assist the CNN model better predict the temperature. Then

CS230: Deep Learning, Autumn 2021, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

we shuffled all the images data (as well as their associated information), randomly sampled 6000 images from all the locations, and resized them to dimensions of $(128, 128, 3)$ as our inputs. After normalizing the input images, we split them into the train, validation and test sets for model training, with 90% of the dataset used as the train set, 7% as the validation set and 3% as the test set.

In the data pre-processing stage for the CNN and LSTM combined model, we filtered for pictures taken at same places between 10 AM and 11 AM and concatenated four consecutive pictures into a group, based on the idea that we can draw information from previous three pictures to facilitate the prediction for the fourth (desired) picture's temperature. We chose to feed in previous three days' pictures, because too many days' pictures would be unnecessary and decrease our model's utility, while too few days' pictures would not provide enough information. Then we transferred the pictures into pixel arrays and performed pixel normalization as before.

4 Methods

Our first method involved using outdoor images to predict the temperature at the time when these images are taken. For the baseline model, we adopted transfer learning by using the InceptionV3 as our pre-trained model, and added some additional layers to output the predicted temperature as a regression task. Among various pre-trained models, we tried VGG16, ResNet50, and InceptionV3. We finally chose the InceptionV3 model because it generated the lowest loss among the three. One possible reason is that the architecture of InceptionV3 allowed us to learn not only features in the large area of the image, but also the area-specific features[6]. Meanwhile, it made an appropriate balance between the accuracy and computation time. Since we believed that the layers in the InceptionV3 model could draw useful information and architecture from the images, we chose to freeze all other parameters in the InceptionV3, and added 2 sets of trainable layers (Conv layer + MAXPOOL) to generate the final temperature prediction. In the dense layer, we chose to concatenate the latitude, longitude and the specific month that each image is taken into the flattened layer of CNN. This was because that based on previous work[7] and also our own experiment results, we found that images alone could be misleading to the model. Two images with similar visual characteristics may have very different temperature, so we decided to provide more background information to the Neural Network to assist the temperature prediction task. When training the model, we used Tanh and ReLU as our activation functions and no activation in our last layer to accomplish the regression task. We employed the Mean Squared Error as the loss function for the model and the RMSprop as the optimization algorithm. Different number of epochs, such as 10, 20, 30 epochs were tested to train the neural network, and most of the hyper-parameters we chose are based on test results of different combinations by careful design. For instance, we used log-based scale in choosing the learning rate.

Our current model structure is shown in the Table 1 below.

Input(shape=(128, 128, 3))
InceptionV3
CONV(32,3,3)
MaxPool(2,2)
CONV(32,3,3)
MaxPool(2,2)
Flatten
Dense(256)
Concatenation with other information
Dense(128)
Dense(64)
Dense(1)

Table 1: First model Structure

Our second model involved the CNN and LSTM combined architecture to predict the temperature at which the last day's picture was taken, given the pictures of previous 3 days. CNN is used to process image information while the many-to-one LSTM is used to add time-series components into our model. As mentioned in the data pre-processing step, we concatenated 4 days' images together,

so our dimension of input data increases by 1 with the value of 4. In this part, we didn't use the pre-trained InceptionV3 model, because it was very computational expensive and not very applicable to 4 dimensional input data. Thus, we trained a CNN model with similar structure as our additional layers in the first model. We also used 2 sets of layers (Conv layer MAXPOOL), a dense layer to 256 units followed by a concatenation of geographical information and month of each image. This output went through another two dense layers, followed by a LSTM layer with 32 unit and eventually into a dense layer with 1 unit of output. Except for the last dense layer, all previous layers are wrapped in a TimeDistributed function so that the model treated 4 images as a group at a time. With the same activation function, epochs, loss function and optimization algorithm as the first model, we achieved better results with this model. The second model structure is shown in Table 2.

Input(shape=(4, 128, 128, 3))
TimeDistributed(CONV(32,3,3))
TimeDistributed(MaxPool(2,2))
TimeDistributed(CONV(32,3,3))
TimeDistributed(MaxPool(2,2))
TimeDistributed(Flatten)
TimeDistributed(Dense(256))
Concatenation with other information
TimeDistributed(Dense(128))
TimeDistributed(Dense(64))
LSTM(32)
Dense(1)

Table 2: Second model Structure

5 Results

This project was divided into two major parts. By leveraging the power of robust deep learning algorithms, we first tried to predict the temperature given a single picture. The second goal was to predict the temperature in the last day by feeding in images from previous 3 consecutive days. Specifically, the Mean Squared Error was used as evaluation metric for both tasks.

In the first task, we predicted the temperature based solely on a single input image. However, we found that the test accuracy was extremely undesirable. Intuitively, images can be very similar in terms of clearness and lighting even if they are taken under very different temperature, so the prediction based on images alone can be easily biased. Hence, we decided to feed more information into the network by concatenate longitude, latitude, and month information into the dense layer. With this structure, we found that both the train MSE and validation MSE could gradually decrease to a level of 25. Below (Figure 1) is a typical loss decaying graph, which is an example with 30 epochs, final train MSE of 26.39, validation MSE of 25.61, and test MSE of 26.71. Most of our predictions can generally capture the proper temperature range from a single image, but at the same time, some predictions can be even 10 degrees away from the true temperature. One closely predicted example is shown in Figure 2, and a corresponding visualization of one early convolutional layer is presented in Figure 3. We can see that different filters have different functions, such as outlining the sky, the cloud, or the buildings. It is also very clear that some filters are more sensitive to vertical edges, whereas others are sensitive to horizontal edges. Such visualizations help us better understand the functions of different filters and also relevance across layers in our network.

In our second model, we fed in 4 consecutive days' images in order to make a better prediction on the last day's temperature. By adding the time-series component through LSTM, we were able to achieve an even lower MSE to the level of 20. With 30 epochs, we can get train MSE of 5.21, validation MSE of 23.89, and test MSE of 20.72. This is reasonable since feeding in more images before the predicted day at a specific location should provide more information and a baseline temperature for the model to give a more accurate prediction. Notice that the train error here is around 5. This is mainly due to the limitations of our datasets, which are taken at around 50 fixed locations only. Therefore, we believe that trying richer image datasets may help reduce the test error for our second task. 5 predicted and true temperature examples from this second model are shown below in Table

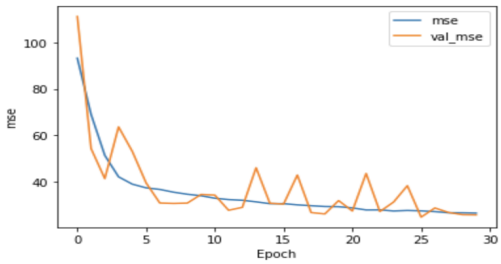

Figure 1: An Example of Loss Pattern During the Model Training with 30 Epochs

Figure 2: Example Image with True Temperature of $15°C$ and First Model Predicted Temperature of $13.7°C$

3. From the results we can see this model gives quite accurate predictions, especially when the temperature is higher, i.e. higher than 15 Celsius degrees. When we were visualizing the middle layers of our CNN model, we found that it was able to capture some details of the picture, such as the continuous strong sunlight that gives a sharp contrast of the objects. These meant our model was able to capture these important details to give a reliable prediction most of the time. However, the predictions were not very accurate when the image is taken in cold weather. We performed some error analysis and found the difference between the predicted and true temperatures was quite big because there is usually no snow or only melting snow in the training images, so the sunlight can still be quite strong with blue sky. Thus, it's understandable that our model cannot give very precise temperatures with limitations in the training datasets. One example is given in Figure 4.

True Temperature	Predicted Temperature
22.0	23.3
8.0	11.4
-12.0	-2.6
17.2	11.9
8.0	9.0

Table 3: 5 Sample Predictions with True Temperature for the Second Model

6 Conclusion

In this project, we utilized deep learning models to predict the temperature of a single image and the temperature of the last day's image given previous 3 days' images. CNN with pre-trained model and our additional layers is used to complete the first task, while CNN + LSTM is used to complete the second task. In the end, we achieved a test MSE around 25 for the first task, and a test MSE around 20 for the second one. Our modeling process indeed proves that giving more information, especially previous days' images, the prediction error can be significantly decreased.

The main challenge involved in this project is that the location and time in a day may impose strong influence on the look of an image, especially on visibility and direction of lighting. Thus, the true temperature at location where the image was taken may be too subtle to be discerned with visual

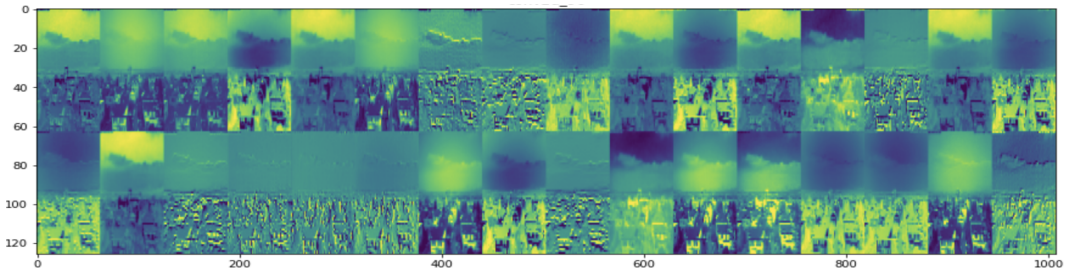

Figure 3: Visualization of Activation Function in a Convolutional Layer

Figure 4: Example Sequential Images with Last Day's True Temperature of -12.0°C and Second Model Predicted Temperature of -2.6°C

characteristics alone. In addition, with limited camera locations, images in our datasets may depict highly similar landscapes and display strong correlations among them. Training on these selected images at limited locations for our models may be affected. Moreover, the nature of our multi-stage model requires high accuracy at the very first stage. We cannot possibly obtain a high accuracy for the subsequent day if our previous days' temperature deviate greatly from the true temperature.

This project can be further improved from several perspectives. First, it may be more ideal to give a precise prediction without other relevant information such as geographical and temporal information that we currently include in our model. Second, more systematic and scientific choices of parameters may improve the model further. Currently, we choose 4 days' images that are taken between 10 AM and 11 AM based on our experience, hence it may be better to do more experiments to choose the number of days and the exact time slots. Third, it may worth exploring the exact features in the image that contribute most in the prediction process. For example, future work can be done to test whether the sky, sunlight, or trees are given the highest weights during image feature extraction and prediction. These can help us understand the underlying logic and reason behind our models.

7 Contributions

All team members contributed equally in this project. We finished the coding and report together. We revised and optimized each other's works. In the end, we combined our findings together and wrote up this paper.

References

[1] Glasner, D. & Fua, P. & Zickler, T. & Zelnik-Manor, L. (2015) Hot or not: Exploring correlations between appearance and temperature. *The IEEE International Conference on Computer Vision (ICCV)*, pp. 3997-4005.

[2] Volokitin, A. & Timofte, R. & Van Gool, L. (2016) Deep Features or Not: Temperature and Time Prediction in Outdoor Scenes. *2016 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1136-1144.

[3] Lu, C. & Lin, D. & Jia, J & Tang, C. (2017) Two-Class Weather Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 2510-2524.

[4] Zhou, H. & Gao, B. & Wu, J. (2017) Sunrise or sunset: Selective comparison learning for subtle attribute recognition. *British Machine Vision Conference*.

[5] Mihail, R. P. & Workman, S. & Bessinger, Z. & Jacobs, N. (2016) Sky segmentation in the wild: An empirical study. *IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 1-6.

[6] Szegedy, C. & Vanhoucke, V. & Ioffe, S. & Shlens, J. & Wojna, Z. (2016) Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2818-2826.

[7] Chu, W. & Ho, K. & Borji, A. (2018) Visual Weather Temperature Prediction. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 234-241.