Particle Detection and Prediction using CNNs and LSTMs

Sophia Wagner
University of Alberta
sjwagner@ualberta.ca

Youssef Ait Haddou
University of Alberta
aithaddo@ualberta.ca

Rajkumar Patel
University of Alberta
rdp1@ualberta.ca

Abstract

Particle tracking is a very powerful tool that plays a crucial role in many research areas. As particles have constant movement and very small sizes, tracking them is quite difficult in general. Because of recent advances in deep-learning techniques, we will be able to track particles optimally with the help of these techniques. In this paper, we propose a deep-learning based method i.e. we will use Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) networks to detect and predict the movement of particle from one frame to another. Our code is available at https://github.com/sophiajwagner/particle-detection-and-prediction

1. Introduction

For a long time in research areas such as biology, physics, etc., time-lapsed microscopy images have been extracted frame by frame from microscopy videos in order to observe the activities of bacteria, viruses, etc. The need for observing the target (i.e., bacteria, virus, etc.) is for the researcher to understand the target or to design a drug for it. In order to observe the activities of such targets, tracking of the particles is done i.e., the location and shape of the particle in each microscopy image is recorded, and then the movement of this particle is tracked from the first image to the last one. Over the years, many particle tracking methods have been developed and significant progress have been made to these methods to achieve full automation. However, the issue of image quality and particle visibility will always be an issue because of photo-bleaching, blurred image and photo-toxicity. This could lead to incomplete or inconsistent footage of particles. Parameter optimization can be done but it is time consuming and requires constant human interaction and guidance. Another drawback of parameter optimization is that new data needs to be optimised accordingly meaning optimisation used in previous data may not be effective in new data. Therefore, in order to deal with the issue of missing data, first we will use the machine learning model proposed by [2], then, we will also be using Convolutional Neural Networks (CNNs) and Long Short-Term Memory Networks (LSTMs) to predict the future movement of the particles with relative accuracy. In order to test the accuracy of our prediction, we will be cutting off final part of the microscopy video and compare our predicted output with the actual output of the data.

2. Data Generation

For data generation, two methods have been developed. The first method comprises of video simulation software in which a large number of conditions are set by the user which are mostly found in particle videos. In this method, we generate frame by frame data from the video simulation software where conditions such as number of frames, standard deviation of background noise, size of particle spots, background pattern, number of particles, etc. are set according to the needs. In the second method, we use random particle video and convert that video into greyscale images frame by frame. As for our model, we will use method 1 since the data generated is much clearer as the parameters are set by the user. These parameters remove all the unnecessary information not required for our model compared to the data generated from method 2 containing the unnecessary information.

3. Method

This work consists of two parts: detecting particles and predicting future movements of particles. Both tasks use Convolutional Neural Networks to capture image features and Long Short-Term Memory to pass information over multiple image frames.

3.1. Convolutional Neural Networks and Long Short-Term Memory

Convolutional Neural Networks (CNNs) take multi-dimensional data, in this case images, as input and output three-dimensional data, where one dimension corresponds to different channels and two dimensions correspond to the input dimensions. For generating the output, one $k \times k$ kernel per output channel slides over the input. The kernel consists of different weights w_i and a bias term b. The output is then a weighted sum of the considered $k \times k$ input pixels plus the bias term. Afterwards, an activation function is used in order to add nonlinearity to the output. This procedure can be repeated in several layers. Whereas the first layer usually captures very basic features of the input like edges, layers at the end of the network can capture more complex shapes of the input.

As the image frames of a video are dependent on each other, Long Short-Term Memory (LSTM) [1] is used, as proposed in [3, 6]. LSTMs model sequences of data. Each part of the input sequence is fed into a LSTM cell. The cell then takes the current part as well as the output of the previous parts of the sequence as input. A LSTM cell comprises an input gate unit and an output gate unit in order to decide which inputs to forget or to keep. The weights of a LSTM cell is shared over the whole sequence of data.

3.2. Particle Detection

For particle detection, the model has to decide for each pixel of the input image whether it is a particle or not. For this task two different models are used: A basic CNN and a combination of CNN and LSTM for comparison.

CNN model architecture. The first model looks at each video frame independently. It follows the architecture proposed in [2]. It consists of three layers: A first layer with 12 channels, a second layer with 32 channels and a third layer with two output channels. The first two layers are each followed by a Leaky ReLU activation function. The last layer is followed by a Softmax activation function to ensure the output corresponds to a probability of whether the considered pixel is a particle or not. The architecture is visualized in figure 1.

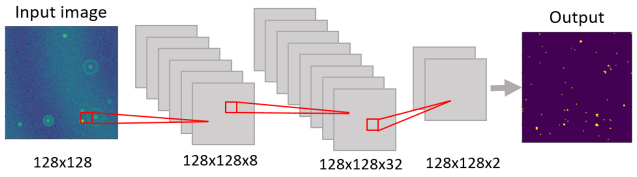

Figure 1. CNN model for particle detection

CNN-LSTM model architecture. The CNN-LSTM model follows a similar architecture consisting of three ConvLSTM layers: 12 channels in the first layer, due to computational reasons only 16 channels in the second layer and two output channels in the third layer. ConvLSTM layers work like LSTMs but with both input and recurrent transformations being convolutional. As common for LSTMs, tanh activation function is used and for the last layer a Softmax function is used again in order to ensure that the output is a valid probability distribution. For the input the whole sequence of particle movement is used and the output is the corresponding sequence giving the probability of being a particle or not for each pixel. Using a LSTM makes sure particle position predictions are depending on previously detected particle positions.

Loss function. Both models used Adam optimization and a weighted cross-entropy as loss function. The loss function is defined the following:

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N}\beta y_i log(\hat{y_i}) + (1-\beta)(1-y_i)log(1-\hat{y_i}),$$

where N is the number of pixels, y_i is tells if the pixel i is a particle or not, $\hat{y}_i$ is the probability predicted by the model that pixel i is a particle. As the data is highly imbalanced, i.e. there are far more pixels non-particles than particles, a weighted loss function is used. β corresponds to the class weight of particles, whereas $1-\beta$ is the class weight of non-particles. The training dataset for this task consists of 20 videos, each with 10 frames, and the validation dataset consists of 3 videos. Due to computational reasons the dataset is very small.

Evaluation metrics. To compare both models different evaluation metrics are used. Accuracy gives the percentage of correctly predicted pixels. The data is highly imbalanced, i.e. there are far more non-particle pixels than particle pixels. Therefore, accuracy is not suitable as always predicting non-particles would already lead to a high accuracy. Therefore, other metrics like Precision and Recall are considered. Recall is the True Positive Rate $\frac{TP}{P}$, where TP is the number of correctly classified positive samples and P the number of positive samples. In other words it tells how many of all the particle-pixels could be correctly identified as particles. Precision is $\frac{TP}{PP}$, i.e. it tells how many of all particles that are predicted as particles are actually particles.

3.3. Particle Prediction

For position forecasting, our model will predict the future coordinates of particles. To do so, the results of the previous parts will be used as sequential data to predict future positions of particles. Since this problem involve sequenced data and can be seen as time series analysis, Long Short-Term Memory (LSTM) networks will be used. Also dropout technique is utilized to improve the algorithm.

LSTM model architecture. LSTM is an extension of the Recurrent Neural Networks (RNN) who were proposed by [4]. This architecture uses cycling of data over loops, allowing output from some nodes to affect the input to the same nodes. However, since this type of architectures can face a problem of vanishing or exploding gradients because it memorizes and reuses every single piece of information. On the other hand, LSTM introduces the notion of selected and forgotten memories. That is, these cells only remembers the important and memorable information which proven to be very efficient. The model is formed with layers of LSTM cells 2. The model here is consisting of two LSTM layers of 128 and 64 neurons respectively,

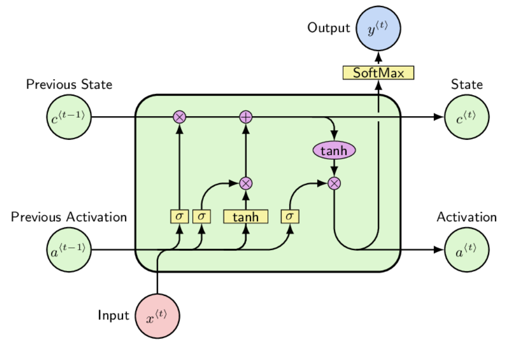

Figure 2. LSTM cell

Dropout. Dropout is technique that used for most Neural Networks to prevent over-fitting of the model. The idea of dropping neurons was introduced by [5]. The idea is to temporarily remove neurons from the network, creating a new architecture which prevents the model on relying on certain nodes as they might be removed, and thus making the model more independent on the input.

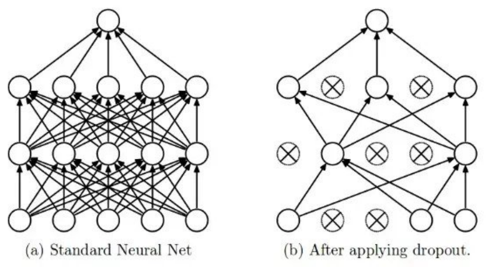

Figure 3. Dropout effect on Neural Network

In our model we use $p = 0.2$, meaning that every node has a probability of 20% to be temporarily dropped from the network.

Flatten and Dense layers. Finally, the out of the previous layers is reshaped (flattened) to a single vector, which is then fed to a final neuron (Dense layer of dimension 1) to provide the final output (a vector of dimension 1).

Loss function. Since this is a regression problem, loss function is a mean squared error defined as:

$$L_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y_i})^2$$

where N is the number of number of data points, y_i and $\hat{y_i}$ are the real and predicted values respectively.

Evaluation metrics. The model is used to predict both x and y coordinates of the particles, and the mean squared error is the metric used to evaluate the results.

4. Results

4.1. Particle Detection

Results for particle detection were produced, as described in the previous section, using a CNN looking at each image frame independently and a CNN-LSTM, that takes the whole image sequence as an input. Table 4.1 shows the corresponding Accuracy, Recall and Precision. Even though the CNN does not consider previous images of a sequence, it could achieve higher accuracy and precision. The CNN-LSTM achieved a higher Recall. Tuning the weights in the cross-entropy loss function, would probably lead to similar results for both models.

Model	Accuracy	Recall	Precision
CNN	0.9940	0.5972	0.2956
CNN-LSTM	0.9768	0.8797	0.1442

Table 1. Results of particle detection

The figures 4 and 7 show the input, output of the CNN and CNN-LSTM model and the ground truth. As the high recall and low precision of the first model confirm, the CNN detects more particles than there actually are. The low recall and high precision of the second model coincide with the results as well. The model detects less particles as there are. Nevertheless, the overall detected positions of particles match with the ground truth for both models.

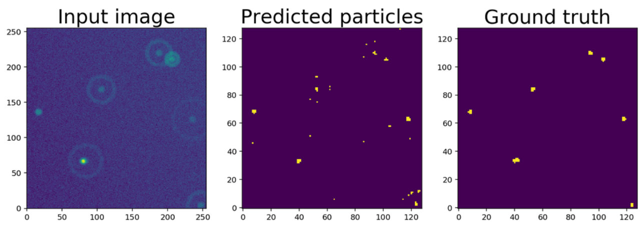

Figure 4. Input, output of the CNN model and ground truth for particle detection

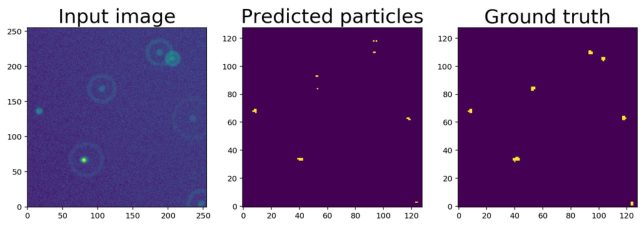

Figure 5. Input, output of the CNN-LSTM model and ground truth for particle detection

As both models achieve similar results, it is probably more reasonable to use a CNN only, as it has a lot less trainable parameters (4,186) than the CNN-LSTM (23,160) and therefore requires less computational power.

4.2. Particle Prediction

The particle position prediction show decent results. For the X-coordinates, the model produced results with Mean Squared error of 24.16%

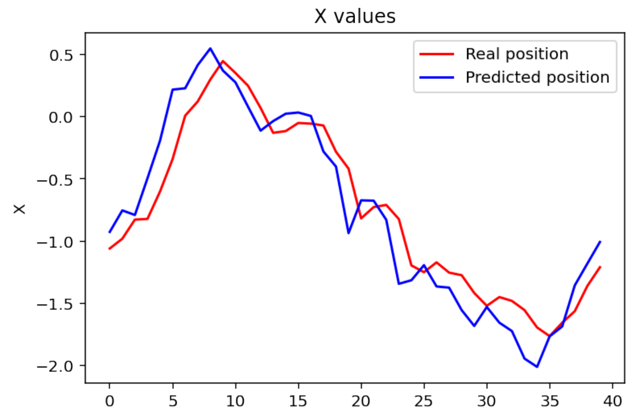

Figure 6. Predicted vs Observed X-positions

For the Y-coordinates, the model gives results with Mean Squared error of 18.71%

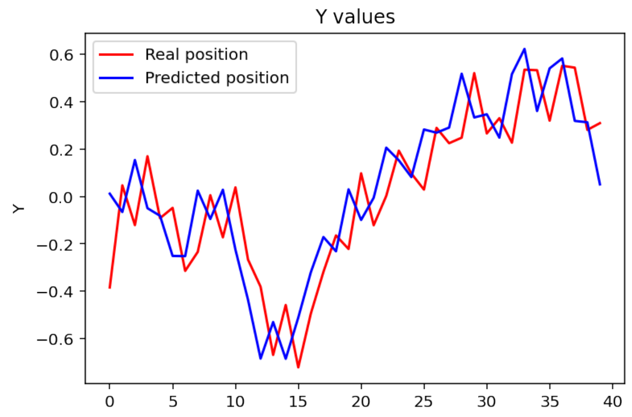

Figure 7. Predicted vs Observed Y-positions

We see that results are decent with MSE of around 20%. In particular, Y-coordinates provide better results because the model was retrained again, giving it more repetions to produce better results.

5. Critique of the model

Particle Detection Both models used for the particle detection could detect the main spots of the particles. However, while the CNN model was predicting too many pixels as particles, the CNN-LSTM model could not detect all particle pixels. Depending on the application you can change the weights in the loss function for higher Recall or higher Precision. It depends on the question if incorrectly identifying too many particles is better than not detecting some particles or the other way round. The overall results could furthermore be improved by using a more complex architecture with more layers as well as using more training data. Due to lacking computational power these improvements could not be conducted.

Position prediction The LSTM model provides good results, however, the accuracy can be improved. That is, we can see that Mean Squared Error on the training data is very low and reaches 6%. This might be an indication of over fitting because of the low error rates and high variance. To improve the algorithm, regularization techniques can be used, in particular increasing the dropout probability or decreasing the number of nodes in the Network might give even better results.

References

[1] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. 2

[2] Jay M. Newby, Alison M. Schaefer, Phoebe T. Lee, M. Gregory Forest, and Samuel K. Lai. Convolutional neural networks automate detection for tracking of submicron-scale particles in 2d and 3d. *Proceedings of the National Academy of Sciences*, 115(36):9026–9031, aug 2018. 1, 2

[3] C. Ritter, R. Spilger, J.-Y. Lee, R. Bartenschlager, and K. Rohr. Deep learning for particle detection and tracking in fluorescence microscopy images. In *2021 IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 873–876, 2021. 2

[4] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986. 2

[5] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 3

[6] Yao Yao, Ihor Smal, Ilya Grigoriev, Anna Akhmanova, and Erik Meijering. Deep-learning method for data association in particle tracking. *Bioinformatics*, 36(19):4935–4941, 07 2020. 2