



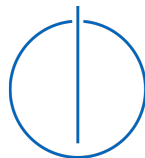
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Guided Research in Informatics

**Learning Aerosol Features From Image
Data**

Rohan Krishna Saxena





DEPARTMENT OF INFORMATICS

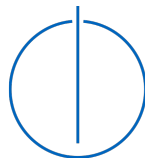
TECHNISCHE UNIVERSITÄT MÜNCHEN

Guided Research in Informatics

Learning Aerosol Features From Image Data

Lernen von Aerosoleigenschaften aus Bilddaten

Author:	Rohan Krishna Saxena
Supervisor:	Prof. Dr. Christian Mendl
Advisor:	Dr. Felix Dietrich
Submission Date:	21st March 2022



I confirm that this guided research in informatics is my own work and I have documented all sources and material used.

Munich, 21st March 2022

A handwritten signature in blue ink, appearing to read 'RKSaxena', with a long horizontal stroke extending to the right.

Rohan Krishna Saxena

Abstract

Learning Aerosol Features from Image Data

This project is aimed at extracting information related to the spread of aerosols directly from image data. This application is particularly interesting in monitoring the spread of infections through air. We can track the growth of the aerosol cloud through time and obtain useful information.

Obtaining image data by performing experiments with human subjects in a controlled experiment is a challenging task. We instead rely on simulations which offer us greater control over the aerosol parameters, as well as providing metadata automatically. These simulations can be generated using a 3D rendering software, like Blender.

Once the image and ground truth data has been obtained, we train a U-Net, which is an encoder-decoder based architecture for image segmentation. It uses convolutions, transposed convolutions, and max pooling layers, along with skip connections. We can train the U-Net to focus only on the aerosol cloud and not on the human subject or other objects in the image. Using the segmented image, we can track the aerosol temporally and extract features.

Contents

Abstract	iii
1. Introduction	1
2. Generating Synthetic Dataset Using Blender	3
2.1. Obtaining Human Images for Simulations	3
2.2. Setting up the Scene in Blender	3
2.3. Creating the Simulations and Exporting Images	4
3. U-Net Architecture	6
3.1. Convolutional Neural Networks	6
3.1.1. Need for CNNs	6
3.1.2. The Convolution Operator	7
3.1.3. Max Pooling Layer	7
3.1.4. Transposed Convolutions	8
3.2. Architecture of U-Net	8
3.3. Application in Image Segmentation	10
3.4. Data Preparation and Augmentation	10
4. Experiments and Results	12
4.1. Evaluation Metrics	12
4.1.1. Pixel Accuracy	12
4.1.2. Dice Coefficient	13
4.2. Experiment 1	13
4.3. Experiment 2	13
4.4. Experiment 3	14
4.5. Experiment 4	14
4.6. Experiment 5	14
4.7. Study of Aerosol Cloud Growth with Time	15
5. Conclusion	18

Contents

A. Software and Libraries Used	19
A.1. Simulation Software	19
A.2. IDEs	19
A.3. Python Libraries	19
 List of Figures	 20
 List of Tables	 21
 Bibliography	 22

1. Introduction

Understanding aerosol dynamics is important for several applications, including the spread of infectious air-borne diseases, but also for fuel injection in combustion engines or airflow through jet engines. In most of these applications, the dynamics cannot be observed on a microscopic (aerosol) level, but only on a relatively coarse, macroscopic level. This is often done by rapid imaging of a controlled experiment, for example, a human exhaling white, illuminated smoke in an otherwise dark room. Aerosol dynamics can be described by (partial) differential equations that are well known, but relatively challenging to integrate numerically. Several additional effects need to be taken into account, such as aerosol-air and gas-fluid-structure interactions (the aerosol interacts with the surrounding air, which is also influenced by the geometry of a face shield, for example) and head pose of the person.

This project is aimed at learning aerosol dynamics directly from image data. The image dataset can be obtained from breathing experiments with humans, studying the effectiveness of masks and face shields. This is a very challenging task. The lack of a large amount of (labelled) training data means (crude) numerical simulations of the aerosol have to be performed as well, to construct an abundance of synthetic training data. We can create our synthetic dataset using a 3D rendering software like Blender, which will provide us metadata like pose and ground truth labels out of the box. As the imaging of breathing out aerosols is typically quite coarse and focused on the entire cloud, individual aerosol dynamics probably cannot be extracted. Instead, the focus of the project is on the dynamics of the volume and extent of the entire cloud. We can track the growth or decay of the aerosol through time. Later, the dynamics can be modelled either with ordinary or stochastic differential equations, which can be learned using neural networks.

We will begin by creating several aerosol simulations in Blender, which will help us generate a large synthetic dataset. Variability in the simulations will be introduced by tuning the fluid simulation parameters like horizontal and vertical speed. We will also add turbulence to some simulations to generate different aerosol growth patterns. Additional variability will be introduced by performing the simulations for different people. Blender allows us to generate image masks in parallel, which contain only the

aerosol simulation. The person is not recorded in the image masks.

Once the dataset is obtained, we will train our neural network for performing the image segmentation task. We will train a U-Net [RFB15] for this task. It is a CNN [L+95] based encoder-decoder based architecture. U-Net is known for its impressive performance on binary image segmentation tasks, even when there is a lack of training data. We will then perform several experiments by training our U-Net on different datasets and evaluate its performance on several testing sets. We will use our final trained model to generate the segmentations for unseen images, which we can then use to study the evolution of the aerosol cloud in different scenarios.

In the future, this work can be extended to study the effects of the pose of the person and their head on the aerosol growth. This can be done using existing pose estimation software like MediaPipe¹ or OpenPose² [Cao+18]. Other effects like different camera angles, face coverings (masks, face shields etc.) and types of exhaling (breathing, speaking, singing etc.) can also be studied. Finally, we can learn a numerical model in the form of a neural network that can predict volume and shape dynamics of the aerosol.

¹<https://mediapipe.dev/>

²<https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/index.html>

2. Generating Synthetic Dataset Using Blender

In this chapter, we take a look at the tools and methodologies we have used to generate the synthetic dataset which we will use later to train our model.

2.1. Obtaining Human Images for Simulations

For obtaining a human agnostic model which is able to easily identify the aerosol cloud irrespective of the person in the simulation, we sourced images from 22 different persons with different skin tones, hair style and color, clothes etc. Their images were either obtained along with their consent, or from websites which provide free to use images^{1,2}. Before these images can be used in the simulation, we need to remove the background and retain only the person in the foreground. In figure 2.1, we can see a sample image before and after removing the background.

2.2. Setting up the Scene in Blender

We require three basic components in our Blender scene to setup the simulation. The first is an image placeholder, which will allow us to reuse the same simulation for different persons. The second component is a smoke flow object, which controls the physics behind generating the aerosol. We can tune several parameters which determine the smoke behavior like smoke density, color, geometry, texture etc. We will use white smoke and only tune the initial speed in the horizontal and vertical directions. The final component is the turbulence component, using which we can add some noise to the aerosol cloud. Tuning its parameters(strength, size and flow) can help us generate interesting smoke patterns.

¹<https://unsplash.com/>

²<https://www.pexels.com/>



Figure 2.1.: Human image before and after removing the background

2.3. Creating the Simulations and Exporting Images

Once we have setup the scene, we can use Blender’s animation tools to bake the animation with the given parameters. We create 7 different simulations for each person. Each simulation was run for 30 frames, generating 30 images. We ran each simulation for 22 persons, giving a total of $22 \times 7 = 154$ scenarios, and a total of 4620 images. One such scenario is shown in figure 2.2.

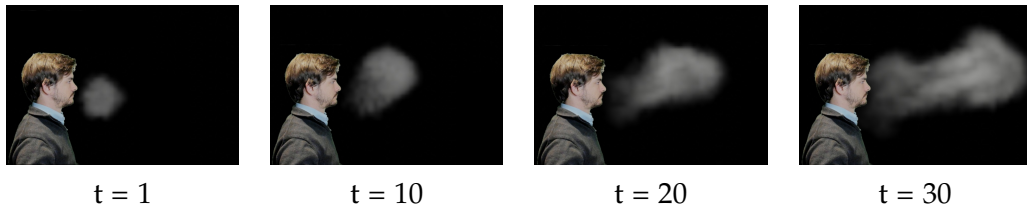


Figure 2.2.: Sample scenario

Apart from the simulation images, we need to generate the image masks as well. These masks will contain only the aerosol cloud and not the person. These masks will serve as the ground truth images when we train our image segmentation model. We separate our smoke flow and turbulence objects and the image placeholder in separate

2. Generating Synthetic Dataset Using Blender

layers, and using Blender's compositor, we can render the two layers in parallel and export the images. In figure 2.3, we can see a few simulation and image masks which were generated in parallel.

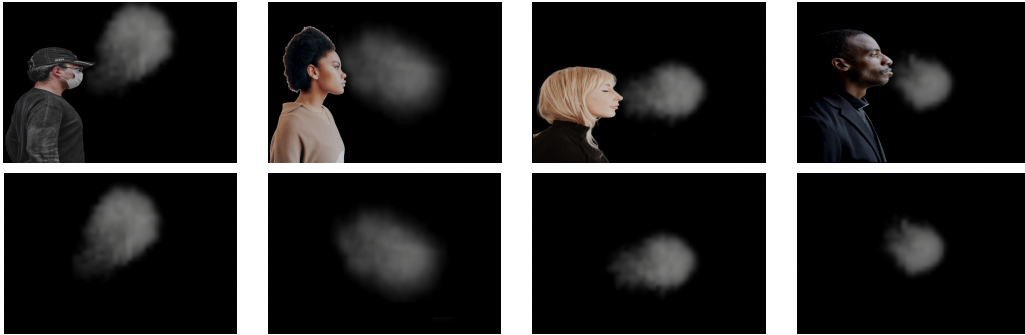


Figure 2.3.: Simulations and corresponding masks

Now that we have our dataset, we can proceed to train our model using different combinations of scenarios as training and test sets, and then evaluate the model performance.

3. U-Net Architecture

In this chapter, we will discuss the U-Net architecture for image segmentation tasks. We will begin by introducing the basic Convolutional Neural Network. Then we will look at the U-Net architecture and the novel concepts it introduced which helped it achieve a significant improvement over other contemporary methods on the ISBI cell tracking challenge 2015¹. Finally, we will see how U-Net is not limited to biomedical image segmentation and can achieve good performance on generic image segmentation problems as well.

3.1. Convolutional Neural Networks

3.1.1. Need for CNNs

Using fully connected neural networks for image related tasks is possible, but it also comes with a few drawbacks:

- Fully connected networks don't scale well for large images. Even tiny images with multiple color channels can result in a large number of parameters, and the size of the network quickly becomes very large and impractical. The huge number of parameters will also lead to overfitting.
- Using FC layers for images does not provide us with any structure.
- Optimizing such networks is also very hard and computationally expensive.
- The network has too many degrees of freedom.

CNNs are often used for tasks like image classification, object localization, object detection and instance segmentation. Convolutional layers provide a more structured approach to the network. Each layer has a specific function for e.g. identifying a certain set features like edges, or more complex features like eyes, nose etc. in case of facial recognition.

¹<http://celltrackingchallenge.net/>

3.1.2. The Convolution Operator

In mathematics, a convolution is an integral that blends one function with another. It expresses the amount of overlap of one function as it is shifted over another function. In the case of a 2D image with one channel, we use filters or kernels which are convolved with the original image in a sliding window fashion, as shown in figure 3.1². The result of the convolution with one filter is a single feature/activation map with a depth of one. In a typical convolution layer, we apply multiple filters with different weights, each producing a single feature map which gives the responses of that filter at every spatial position.

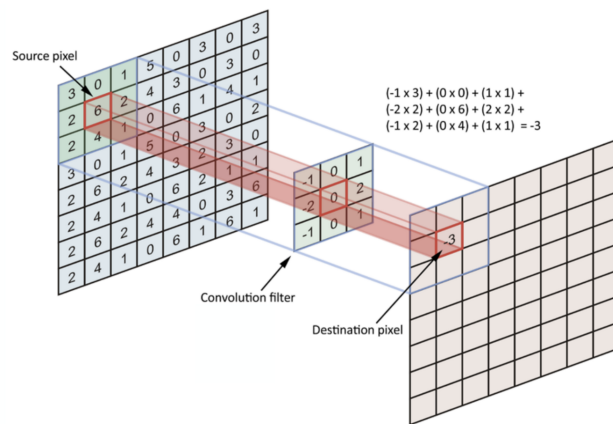


Figure 3.1.: Convolution on a single channel image

Applying several filters allows us to extract different features, and also reduce the spatial size of the image, reducing the number of parameters. These filters are learned via backpropagation.

3.1.3. Max Pooling Layer

Pooling layers progressively reduce the spatial size of the representation to reduce the number of parameters and computation in the network, and hence also control overfitting. They are commonly inserted in between successive convolution layers. The pooling layer operates independently on every depth slice of the input and resizes it spatially. The most common is a max pooling layer with filters of size 2x2 applied with a stride of 2 which downsamples every depth slice input by 2 along both width and

²Image Source: <https://towardsdatascience.com/simple-introduction-to-convolutional-neural-networks-cdf8d3077bac>

height, discarding 75% of the activations. Max pooling operation can be seen in figure 3.2³.

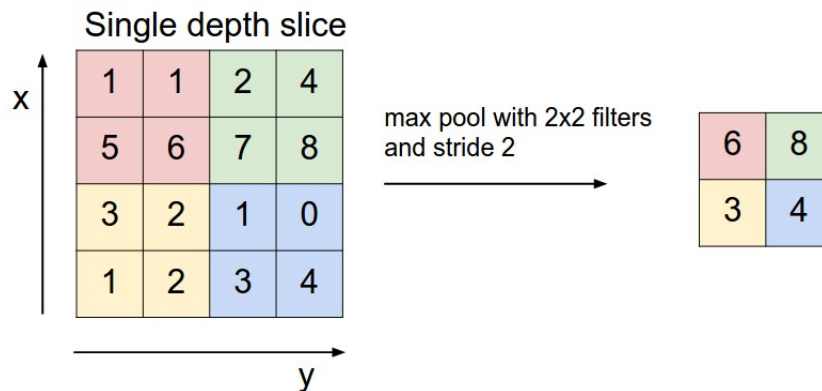


Figure 3.2.: Max Pool operation

3.1.4. Transposed Convolutions

Transposed convolutions can be thought of as the inverse operation of the regular convolution. It allows us to increase the size of the output feature map compared to the input feature map.

3.2. Architecture of U-Net

U-net has an autoencoder like structure i.e., it has an encoder and a decoder part in the architecture, as shown in figure 3.3⁴. The left half is called the contracting path, the right half is called the expansive path and the grey arrows are called the skip connections. We also make use of Batch Normalization, which helps in stable learning.

- **Contracting Path:** In the contracting path, we perform 3x3 convolutions twice (with ReLU activation). This is followed by downsampling through a max pooling layer with filters of size 2x2 and stride 2. We perform this series of operations a total of 4 times. As we progress through the contracting path, the spatial size decreases whereas the number of channels increases. We are essentially reducing the spatial resolution, while increasing the semantic resolution. Successive application of convolution and pooling layers results in an image representation with

³Image Source: <https://cs231n.github.io/convolutional-networks/>

⁴Image Source: <https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/>

3. U-Net Architecture

high semantic resolution and low spatial resolution at the bottom of the U-Net. This gives us a rich representation with higher level information and features about the image, but it lacks spatial information.

- **Expansive Path:** In the expansive part, we perform 3x3 convolutions twice to reduce the number of feature maps, followed by 2x2 transposed convolutions which double the size of the image (height and width dimensions). This helps us in recovering the size of the original image.

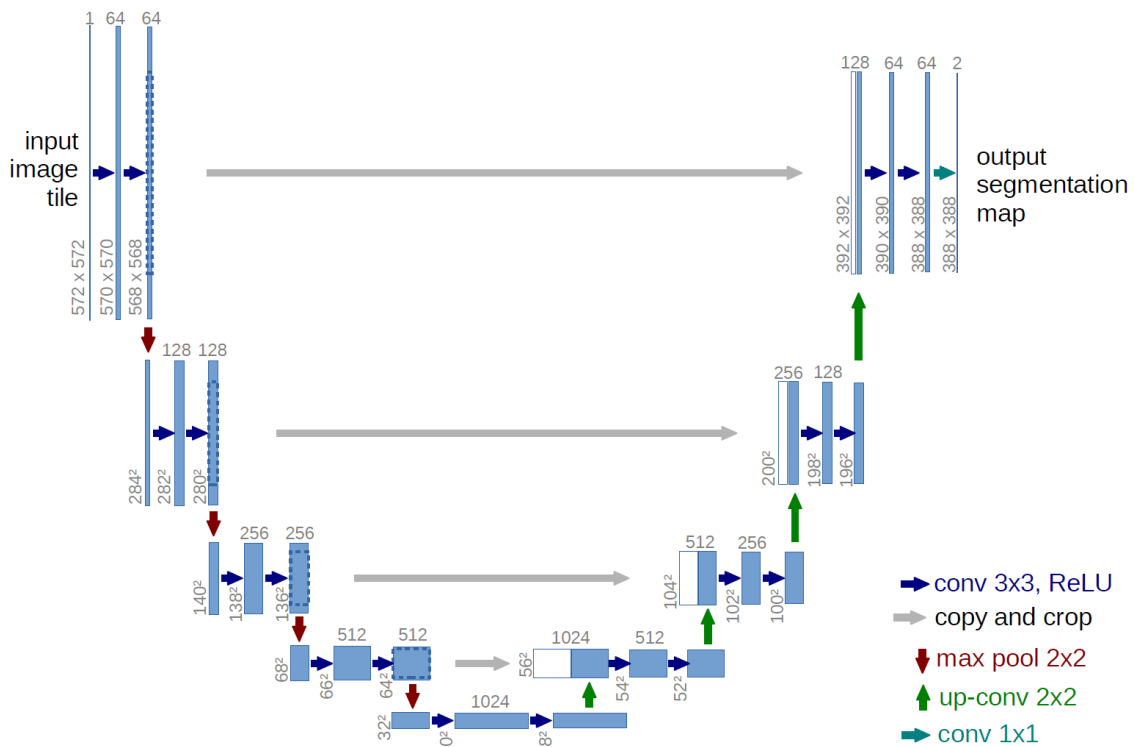


Figure 3.3.: U-net architecture

- **Skip Connections:** The stored information from the encoder part is passed to the decoder by skip connections. These skip connections help us pass spatial information into the deeper layers. The feature maps at the same level of the U-Net in the encoder part (spatial information) are passed to the decoder part, where they are combined with the upsampled output (semantic information) of the previous layer. This concatenation of feature maps helps give localization information.

- **Classification Layer:** In the final step, we have a representation of the image which we need to map to a segmentation map. Since we have a binary classification problem, we will map it to 2 channels. We need to have one channel per class. This is done using a 1x1 convolution layer which is equivalent to a fully connected layer. It acts as a weighted sum over the channels of the input feature map. We will do 2 1x1 convolutions in the final layer which will give us two feature maps as output. One feature map will correspond to the foreground and the other to the background. Finally, we apply a softmax to the final feature map of depth 2, which gives us probability scores for each pixel over all the possible classes (2 in this case).

3.3. Application in Image Segmentation

In the semantic segmentation problem, we have an image as input and then we try to assign each pixel in the image a class label. This is called a segmentation map. It is a kind of classification problem which is applied to each pixel. In our case, we have a binary semantic segmentation problem. Each pixel in the image will be classified as being foreground (part of aerosol) or the background.

U-Net achieves really good performance even with low amounts of data with augmentation. It is able to handle almost touching objects of the same class i.e., can detect fine boundaries between objects of same class.

U-Net has been widely used for pixel-wise image segmentation for medical images, satellite images etc. Its performance can be improved by using a pre-trained encoder, which can be trained on a large data set like ImageNet[IS18]. It has also been used to solve the pansharpening problem of remote sensing images [Yao+18]. U-Net can also be expanded to work with 3D volumetric data to generate dense 3D segmentations from sparsely annotated volumetric images [Çiç+16]. This was done by replacing all 2D operations of the original U-Net by their 3D counterparts.

3.4. Data Preparation and Augmentation

Our original dataset contains RGB images of size 1080 x 1920. We re-scale them to a size of 160 x 240, because of computational constraints. The ground truth masks are of the same dimensions, but contain only one channel. Each pixel here is either 0 or 1.

To add more variability to our dataset, we have used the Albumentations library ⁵ for data augmentation. It provides a rich variety of image transform operations which can be easily applied. We have used image rotations, horizontal flip, and Gaussian blur. These transformations lead to scenarios which are realistic and we can expect to see them in the real world. Figure 3.4 shows some images after applying the transformations.

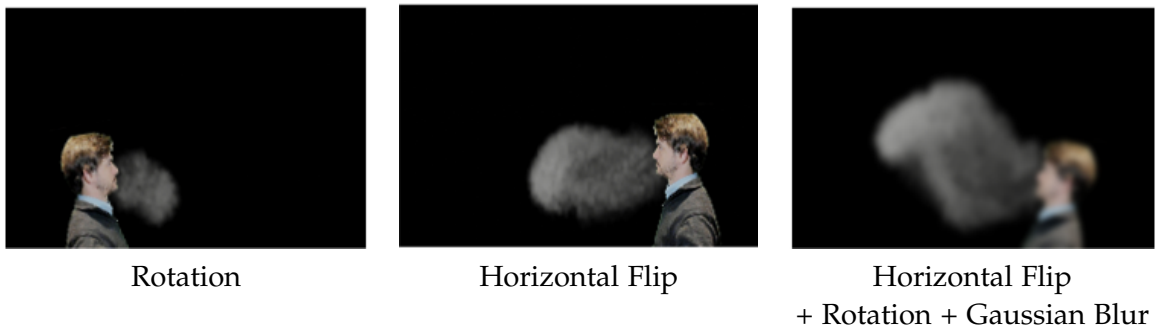


Figure 3.4.: Sample Simulation

⁵<https://albumentations.ai/>

4. Experiments and Results

In this section, we will train several models with different training data, and evaluate their performance on different combinations of test data. Then we will choose the best performing model which generalizes well and use it to identify aerosol clouds on unseen images. Finally, we will see the evolution of the aerosol clouds with respect to time.

4.1. Evaluation Metrics

In this section, we will take a look at the metrics we have used to evaluate the performance of our model on unseen data.

4.1.1. Pixel Accuracy

Pixel accuracy determines the percentage of pixels in the image that are classified correctly. It is conceptually easy to understand, but it is not a very reliable metric. In cases where there is a class imbalance in the image (no. of background pixels \gg no. of foreground pixels), a trivial solution like predicting the background class for every pixel will give a high accuracy. A high pixel accuracy does not always imply a superior segmentation. The formula for pixel accuracy is given below in figure 4.2¹.

$$\text{Accuracy} = \frac{(TP + TN)}{(TP + FP + TN + FN)}$$

Figure 4.1.: Pixel Accuracy formula

¹Image Source: <https://www.mydatamodels.com/learn/how-good-is-your-machine-learning-algorithm/>

4.1.2. Dice Coefficient

Dice Coefficient or Dice Score is defined as twice the area of overlap between the prediction and ground truth image divided by the total number of pixels in both the images. It is a much more reliable and robust measure. Its value ranges from 0 to 1. A value closer to 0 indicates that the overlap between the prediction and ground truth is very small i.e. the prediction is not good. As the value moves closer to 1, the overlap between the prediction and ground truth also increases. Simply put, the closer the value is to 1, the better the prediction. The formula for calculating the Dice Coefficient and the intuition behind it is illustrated in figure 4.3².

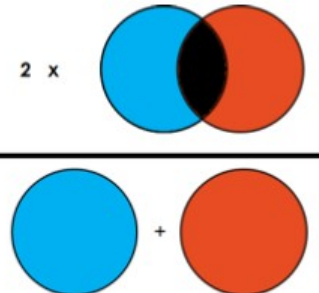
$$DICE = \frac{2 \times (S_g \cap S_t)}{|S_g| + |S_t|} = \frac{2TP}{2TP + FP + FN} = \frac{2 \times \text{Venn Diagram}}{\text{Two Disjoint Circles}}$$


Figure 4.2.: Dice Coefficient Formula

4.2. Experiment 1

In our first experiment, we use a single simulation on a single person. We split our 30 image dataset into 80-10-10 training-validation-test sets. This gives us a 24 images for training, with 3 for validation and 3 for testing. As expected, our model achieves impressive results even with the small amount of training data, since we made use of data augmentation. We achieved a test set accuracy of 99% and a Dice score of 0.98.

4.3. Experiment 2

In our second experiment, we use the model trained in experiment 1 and test it on another simulation of the same person. The test set in this case has 30 images. We observe that our accuracy improves slightly, but there is a decrease in the Dice score.

²Image Source: <https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2>

We achieved an accuracy of 99.6% and a Dice score of 0.93. This experiment indicates that we need to a more diverse training set.

4.4. Experiment 3

In this experiment, we try to understand the generalization power of a model trained on a simulation of just one person. We will use our previous model on a comparatively large testing set containing 2310 images (7 simulations with 11 different persons i.e. 77 different scenarios). As expected, our accuracy and Dice score drop further. We achieve an accuracy of 98.7% and the Dice score drops to 0.88. Our model is not able to generalize to all the scenarios in the testing set. Particularly, in cases where the person is wearing light colored clothing, have grey or blonde hair, or light is reflecting off the person's face, the model gets confused and classifies it as part of the aerosol cloud.

4.5. Experiment 4

In our fourth experiment, we split our previous 2310 image dataset using an 80-10-10 training-validation-testing scheme. The images are sampled randomly for the training, validation and test sets, so we have a good mix of images from different scenario. Training on such a diverse dataset yields good results as we achieve a test set accuracy of 99.9% and a Dice score of 0.992.

4.6. Experiment 5

In our final experiment, we try to understand the generalization power of our model trained on a diverse dataset. The results of this experiment will tell us whether U-Net is appropriate for our particular problem or not. We use our model from experiment 4 and test it on simulations for 11 different persons. These simulations are completely unseen by the model. We achieve virtually the same metrics as we did in the previous experiment, a test set accuracy of 99.9%, and a Dice score of 0.992.

The results of the experiments are compiled in table 4.1. Based on these results, we can confidently say that our U-Net model has been able to generalize well to unseen data. We have trained it on a diverse dataset along with augmentations, and this has allowed our model to identify aerosols in unseen images with good results.

Experiment No.	Accuracy	Dice Score
1	99.01%	0.983
2	99.66%	0.93
3	98.76%	0.879
4	99.93%	0.992
5	99.94%	0.993

Table 4.1.: Results of the experiments

4.7. Study of Aerosol Cloud Growth with Time

Now that we have trained and tested our model, we can use it for the inference stage. In particular, we will take a look at two scenarios to visualize the performance of our model. The predictions made by the model are overlaid over the original image in red. In the first scenario, shown in figure 4.3, we can see that the prediction is able to identify the aerosol and its boundaries.



Figure 4.3.: Model predictions overlaid on original images for Scenario 1

In the second scenario, shown in figure 4.4, the model prediction suffers because of the light colored hair, which is being detected as a separate aerosol cloud. Our model is detecting the original aerosol cloud well, but it is also detecting a false positive cloud, which is actually the hair of the person. But as the simulation proceeds and the size of the aerosol cloud increases, the model puts more emphasis in detecting the ground truth cloud and the size of the false positive cloud decreases, giving better performance.

We can now track the growth of the aerosol cloud in any simulation. We tracked the aerosol growth for the 7 simulations for each person. In figure 4.5, we have plotted the aerosol cloud growth with time for each person.

Since the simulations we have created are same across different persons, we would



Figure 4.4.: Model predictions overlaid on original images for Scenario 2

expect the aerosol cloud growth plots to also be same across persons. Our plots show that it is indeed true, with only minor differences observed. The general trends are pretty similar across all persons.

The first three simulations (1 to 3) had a linear growth of the aerosol cloud, and our plots show that this behaviour is captured by our model. In the rest of the simulations (4 to 7), we added different amounts of turbulence, leading to nonlinear growth of the aerosol cloud. Our model is also able to capture this nonlinear growth as well, as shown in the plots.

4. Experiments and Results

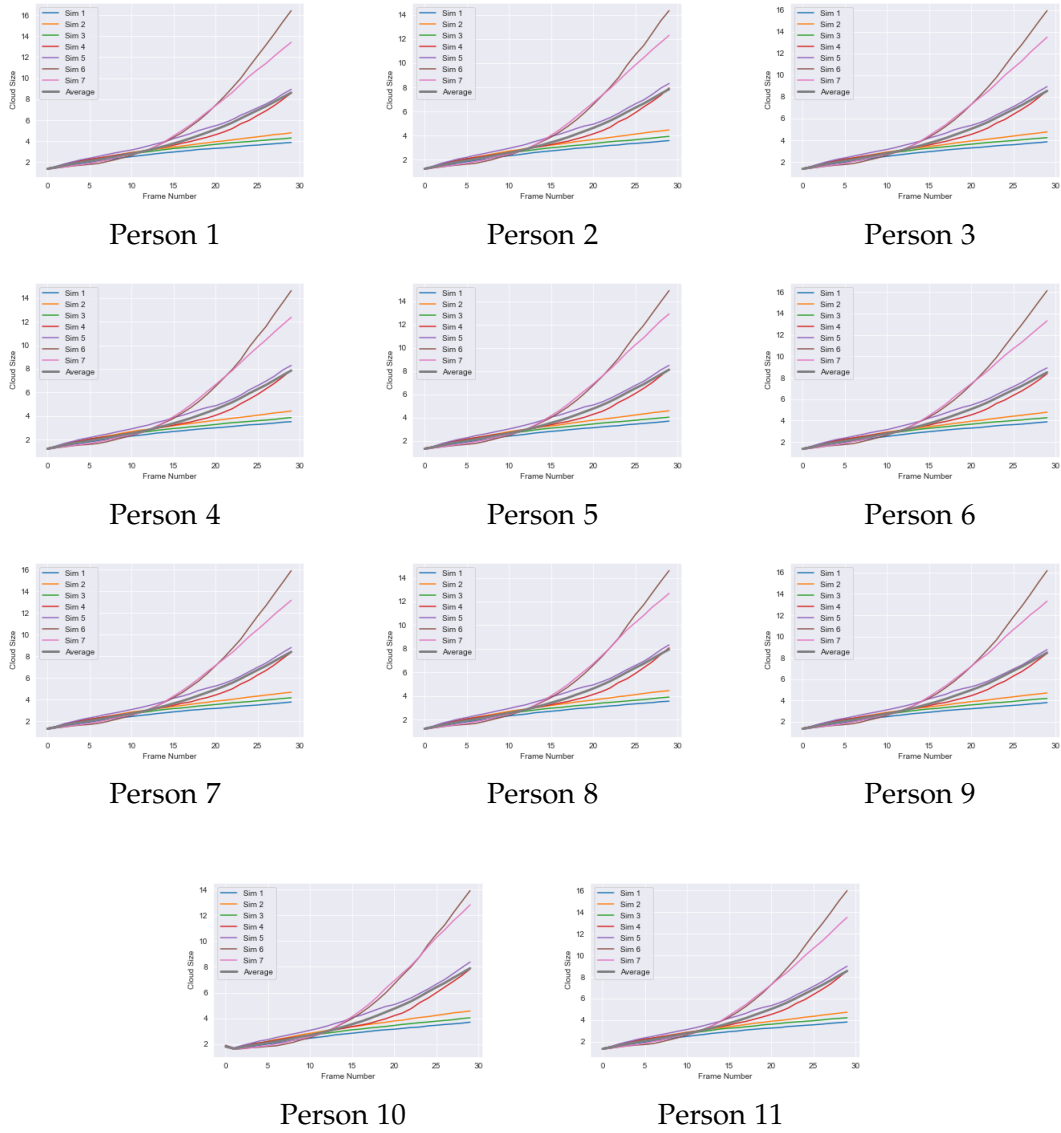


Figure 4.5.: Aerosol cloud growth with time

5. Conclusion

Understanding the dynamics of aerosols and how they behave in the real world has several applications, especially in predicting the spread of air-borne diseases. However, training such a model requires a diverse dataset. Generating synthetic data is a fast growing and popular way for training machine learning models. When compared with the alternative of capturing data from experiments with willing human participants, this method can be very efficient, flexible and cost-effective.

In this project, we have demonstrated that we can create realistic simulations of aerosol spread using Blender. Using a software for this purpose allowed us to simulate several different scenarios, which would not have been possible to do in real life. We subsequently trained a U-Net on the training data, and saw that it generalizes well to unseen data. We were later able to use it to track the growth of aerosol clouds through time across all simulations.

In the future, this work can be extended to include other factors which might influence the aerosol dynamics, like head pose, face coverings, camera angles etc. We can also learn a numerical model which can predict the volume and shape dynamics of an aerosol.

A. Software and Libraries Used

A.1. Simulation Software

Software Name	Version
Blender	2.93

Table A.1.: Simulation software used

A.2. IDEs

IDE Name	Version
Jupyter Notebook	6.4.0
Pycharm	2021.2.3 (Community Edition)

Table A.2.: IDEs used

A.3. Python Libraries

Library Name	Version
Numpy	1.17.0
Pillow	8.3.1
Matplotlib	3.3.4
Pytorch	1.6.0
Torchvision	0.7.0
Albumentations	1.0.3
Tqdm	4.62.3

Table A.3.: Python libraries used

List of Figures

2.1. Human image before and after removing the background	4
2.2. Sample scenario	4
2.3. Simulations and corresponding masks	5
3.1. Convolution on a single channel image	7
3.2. Max Pool operation	8
3.3. U-net architecture	9
3.4. Sample Simulation	11
4.1. Pixel Accuracy formula	12
4.2. Dice Coefficient Formula	13
4.3. Model predictions overlaid on original images for Scenario 1	15
4.4. Model predictions overlaid on original images for Scenario 2	16
4.5. Aerosol cloud growth with time	17

List of Tables

4.1. Results of the experiments	15
A.1. Simulation software used	19
A.2. IDEs used	19
A.3. Python libraries used	19

Bibliography

- [Cao+18] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh. “OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields.” In: *CoRR* abs/1812.08008 (2018). arXiv: 1812.08008.
- [Çiç+16] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. “3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation.” In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*. Ed. by S. Ourselin, L. Joskowicz, M. R. Sabuncu, G. Unal, and W. Wells. Cham: Springer International Publishing, 2016, pp. 424–432. ISBN: 978-3-319-46723-8.
- [IS18] V. Iglovikov and A. Shvets. “TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation.” In: *CoRR* abs/1801.05746 (2018). arXiv: 1801.05746.
- [L+95] Y. LeCun, Y. Bengio, et al. “Convolutional networks for images, speech, and time series.” In: *The handbook of brain theory and neural networks* 3361.10 (1995), p. 1995.
- [RFB15] O. Ronneberger, P. Fischer, and T. Brox. “U-net: Convolutional networks for biomedical image segmentation.” In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.
- [Yao+18] W. Yao, Z. Zeng, C. Lian, and H. Tang. “Pixel-wise regression using U-Net and its application on pansharpening.” In: *Neurocomputing* 312 (2018), pp. 364–371. ISSN: 0925-2312. DOI: <https://doi.org/10.1016/j.neucom.2018.05.103>.