Understanding Crowd Behavior using Unsupervised Deep Neural Networks

Sepehr Janghorbani sj620@rutgers.edu
Shardul Naithani sn533@rutgers.edu
Kshitij Shah ks1223@rutgers.edu
Aravind Sivaramakrishnan as2578@rutgers.edu

Abstract

We propose two unsupervised deep neural network based approaches to model crowds, towards analyzing and predicting their behavior in known or unknown situations. The proposed approaches include non-linear PCA based networks belonging to the autoencoders family, as well as deep generative models trained under an adversarial setting. We present the results of these approaches on our chosen dataset.

1. Introduction

Understanding behavior of crowds (humans, vehicles, animals) is essential for being able to analyze and predict their behavior in known or unknown situations or environments. For instance, we may wish to understand the characteristics of a panic that happens in a stadium and how such event may affect the behavior of the crowd, e.g. the overall movement of the people or the bottlenecks of the map. Moreover, modeling crowd behavior can also be beneficial for making layout and design decisions, like where to place barricades or which gates to be used.

Such analyses pose unique challenges, since crowds are typically represented as trajectories of individual agents, and that different crowds can have different number of agents. A machine learning model that has learned from a crowd of M agents may not necessarily generalize well to a crowd of $K = M + N$ agents. Thus, it is imperative that a suitable, robust representation of the crowd must be constructed before a machine learning model can be applied to them.

In recent years, neural networks have gained prominence due to their successes in building powerful feature representations that can match or exceed human-level performance in tasks such as image recognition [12]. However, their successes in constructing purely unsupervised representations of data, in particular generative modeling, has not matched that of their supervised counterparts.

In this project, we aim to implement novel unsupervised deep learning methods such as variational autoencoders [11] and generative adversarial networks (GANS) [7] for the problem of modeling crowd behavior. We present the results of our proposed models on the crowd dataset used by Alahi et al. [3]. The rest of this report is structured as follows. Section 2 presents related work, and introduces state-of-the-art unsupervised deep learning approaches. In Section 3, we provide a description of the dataset used, as well as a discussion of the different representations that were constructed. In Section 4, we provide an overview of the different models that were trained on the different representations. Section 5 presents the setup of our experiments and the obtained results. A conclusion and summary of our findings are provided in Section 6.

2. Prior Work

Modeling crowd behavior has been a problem studied by the computer vision and computer graphics communities since the early 2000s, and with recent advances in computation, the problem has seen many advances; A comprehensive survey of these approaches can be found in Li et al. [13].

Ali and Shah [3] proposed an algorithm for tracking individuals in high-density crowds. More recently, Zhou et al. [19] proposed a Mixture Model of Dynamic Pedestrian-Agents to learn the collective behavior of large crowds. They used trajectory information collected from video data to predict the movements of individuals. Alahi et al. [2] proposed a socially aware model for forecasting pedestrian destination.

However, most work in the area has focused on purely statistical methods for prediction. Relatively little work has been carried out in applying deep learning methods for understanding crowd behavior, unsupervised or otherwise: Zhang et al. [18] applied deep Convolutional Neural Networks (CNNs) for the task of crowd counting, or counting the number of individual agents present in a crowd, and Lv et al. [14] proposed a deep neural network model for traffic flow predictions.

Of the literature that we surveyed, we found that only

Zhu et al. [20] leveraged a GAN-based architecture for the closely aligned problem of wave-dynamics simulation. The method was able to learn abstract and complex physical rules such as reflection and diffraction from raw images.

3. Dataset & Representations Used

3.1. Dataset

The dataset we used consists of trajectory information collected at a corridor in a subway station [2]. The dataset consists of about 100,000 individual trajectories and about 42 million points, with each point representing an agents location at a specific timestamp. Thus, each individual's movement corresponds to 420 points on the map with a timestamp attached to each one of them, from the time they enter the scene till they leave the scene. The data was collected for two hours every day, over a span of 10 days. The raw data was in the form of comma separated values (.csv files), with each row representing a point. In addition, each point is sampled at an interval of 100 ms, i.e. every agents movement is encoded with 10 points per second. Each point is represented by the following information in the dataset: *Timestamp, Place, X_coord, Y_coord, Track_id*. The place refers to the different corridors that the dataset was originally collected at, whereas we are dealing with only a single corridor.

3.2. Representations

Like most computer vision problems, representation, or feature extraction, is a key challenge for this problem. Using the raw information provided to us as features is unlikely to provide meaningful insights into the behavior of the crowd. Thus, a better approach would be to plot the points on a map, which in this case, is a plain canvas representing the top view of the corridor where data was collected. The input to the model is the map as an image file. We tried three different representations with multiple variations in each one.

3.2.1 Trajectory-based Representation

An intuitive way to represent the data presents itself immediately: plot all the trajectories initiated between time t and $t+\delta$ on a single map. However, this leads to the problem of congestion of trajectories. When too many trajectories are bundled together over a small area, the congestion of points make the map unintelligible. Moreover, these overlapping trajectories obscures crowd density information. We applied two techniques to mitigate this problem. The first was to plot trajectories with semi-transparent overlays, i.e. overlaying trajectories on the map one by one with a predefined transparency index α attached to them. This would allow for darker colors where many trajectories overlap, providing some useful insight about the crowd density. The second technique was to use a unique different color for every trajectory, hence making them more easily distinguishable from one another. Both changes improved the intelligibility of the images for human subjects as shown in Figure 1.

Although, not effective on its own as we will discuss in the results part, these representations paved the way for two more insightful representations. The first one being the density heatmap which focuses on the crowd density information of the dataset and the second one being the agent-based representation which captures the movement of individual agents in the crowd throughout the entire time duration.

3.2.2 Density Heatmap Representation

Unlike the trajectory-based representation, which tries to capture both the movement of individual agents as well as the crowd as a whole, the density heatmap representation focuses only on the distribution of crowd density among different regions in the map, thus making it more effective in representing overall crowdedness of the scene and the behavior of crowd in different areas. These high density regions can be interpreted as the potential bottlenecks of the map, which can obviously be useful in applications such as map design.

The density heatmap representation treats the location of each agent in a probabilistic manner modeled by a unit Gaussian of two dimensions with mean at the provided location of agent and a predefined variance. This allows us to model the fact that an agent in crowd, in our case a human, is not a point agent in the real-world. Moreover, it also mitigates the inevitable error associated with tracking and measurement in the real world by treating each agent as a stochastic process instead of a single point.

For generating density heat maps the following procedure is followed: Every heatmap is initialized as an $m \times n$ zero matrix, where m, n are the height and width of the intended image, respectively. For the heatmap corresponding to the time interval t to $t+\delta$, a unit Gaussian kernel of predefined size and variance is added at the location specified in the data for each point between time t and $t+\delta$. Then, the matrix is normalized using min-max scaling to convert all the values to $[0, 1]$. The final heatmap image is generated by color-coding the density values. This approach does not discriminate between individual agents. The images provided in Figure 2 show sample generated heatmaps. The size of the Gaussian kernel was set to 19 and the variance was set to 3..

3.2.3 Agent-based Representation

Spawning off the trajectory-based representation, This representation focuses on capturing individual agents movements through time; thus modeling the social force and be-

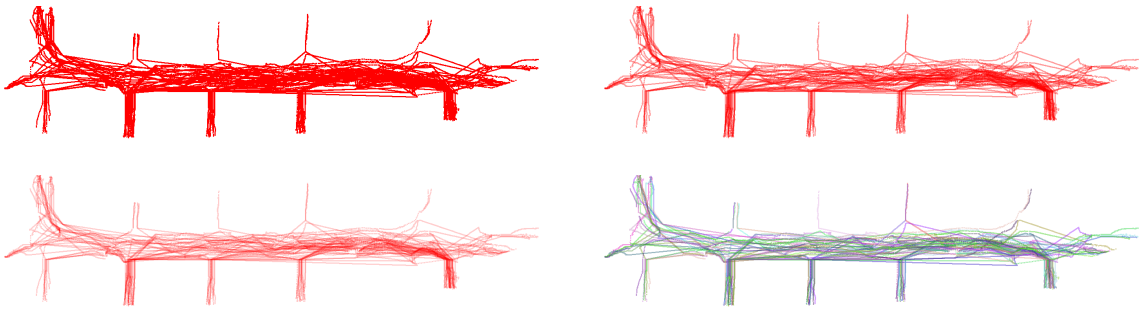

Figure 1: **Trajectory-based representation**: The first three images from top left show the effect of increasing the transparency index α. The image on the bottom right shows the use of random colors for individual trajectories

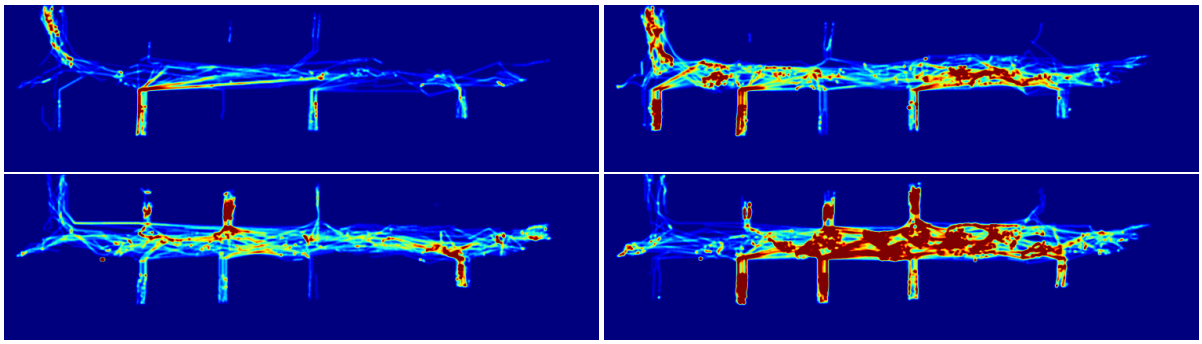

Figure 2: **Density heatmap representation**: Density heatmaps represent the distribution of crowd density on the map. The image in the top left suggest a low crowd density while the one in the bottom right represents a very crowded scene.

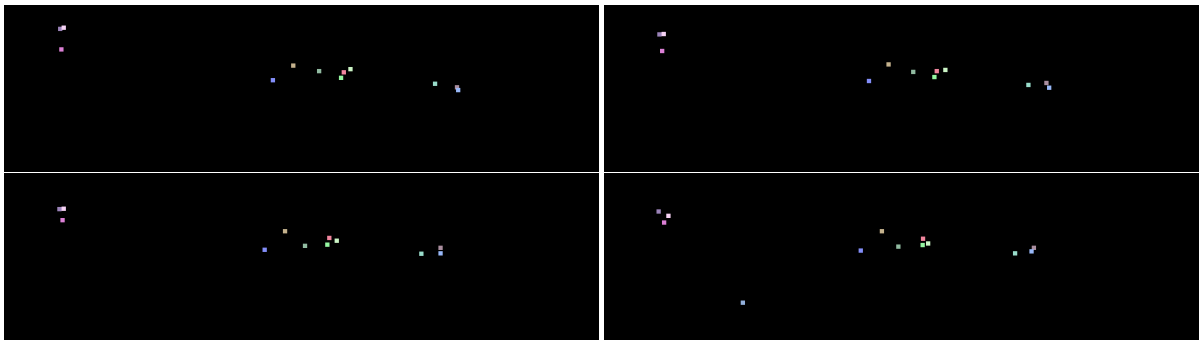

Figure 3: **Agent-based representation**: The four images depicts the position of agents at the interval of one second. The top left being the first one in the sequence and the bottom right being the last one.

havior of crowd as whole since all agents in the scene at time t plotted together.

The procedure for generating this representation is fairly straightforward. Each agent present in the scene at time t is plotted as a point on the image. For practical reasons, the agent is not represented by a single pixel but a $s \times s$ pixel square centered at the position of the agent, where s is a predefined constant. For the next image, agents present in the scene at time $t + \delta$ plotted in the same way. To identify individual agents, each agent is assigned a random color from RGB color-space, which is preserved between all images. A variation of this approach uses same color for every agent. Figure 3 shows images corresponding to this representation.

4. Modeling Approaches

Our approach to the problem of understanding crowd behavior is twofold. In our first approach, we propose to train generative models on the static images of the crowd density heatmaps constructed in Section 3. In our second approach,

we aim to build a robust representation of the movement of agents across time by training a video prediction model on the frames of the agent-based representation.

4.1. Approach I

4.1.1 Variational Autoencoder (VAE)

Given a set of inputs $\mathbf{x} = \{x_1, x_2, ...x_m\}, x_i \in \mathbb{R}^d$, an *autoencoder* [9] is an unsupervised neural network that applies backpropagation, setting the target values to be equal to the inputs, i.e. $y_i = x_i$. It tries to learn a function $h_{W,b}(x) = x$. In other words, the model tries to learn a non-linear approximation of the identity function by minimizing the Root Mean Square Error (RMSE) loss between the input and the corresponding output of the autoencoder.

The standard autoencoder structure consists of two layers of stacked neural networks. The first layer is the *encoder*, which takes as input an image, and *encodes* it into a vector, which we will call the *latent vector*. The *decoder* network takes as input a latent vector, and *decodes* it into a full-sized image.

Though the standard autoencoder structure is useful for practical applications such as data denoising and dimensionality reduction, it is not a traditional generative model in the sense that it cannot create a latent vector other than encoding it from an image. The variational autoencoder [11] resolves this issue by adding a constraint on the encoding network, that forces it to generate latent vectors that approximately follow a standard Gaussian distribution. The decoding network can now generate new images from its learned distribution simply by sampling latent vectors from the unit Gaussian distribution.

4.1.2 Generative Adversarial Networks (GANs)

Consider an initial data distribution $p_{data}(x)$ and a noise prior $p_z(z)$, which we will refer to as the *code*. The generative model generates images $D(z)$ using the code. The discriminative model D and the generative model G are locked in a minimax game against each other, where the discriminative model is trying to maximize the probability score it assigns to images drawn from the original data distribution (denoted by $D(x)$), and minimize the probability score it assigns to images that are generated by G (denoted by $D(G(z))$). Th generative model G is thus trying to do the opposite. Let the value function of this minimax game be denoted by $V(D, G)$. The GAN framework can thus be mathematically represented as:

$$\min_G \max_D V(D,G) = \mathop{\mathbb{E}}_{x \sim p_{data}(x)} [\log D(x)] + \mathop{\mathbb{E}}_{z \sim p_z(z)} [\log(1 - D(G(z)))]$$

Since maximizing probabilities is equivalent to maximizing the log probabilities, we have introduced logarithmic terms into our expectations in the above equations. In practice, this helps us easily compute gradients for the backward pass of the neural network, as the outputs at the final layer of our models can be expressed as an exponential. It is interesting to note that optimizing the above equation is intractable. Thus, an iterative, numerical approach, such as stochastic gradient descent (SGD) is used for training GANs.

4.2. Approach II

Under this approach, we make use of the multi-scale video forecasting model proposed by Mathieu et al. [15]. Though we initially considered the LSTM based autoencoder architecture proposed by Srivatsava et al. [17], our initial experiments with the architecture showed that the L2 loss used for training it inherently produced blurry frames. This phenomenon can be attributed to the fact that the usage of the L2 loss comes from the assumption that the data is drawn from a Gaussian distribution, and thus will perform poorly on multimodal distributions. We wanted to emphasize that our goal is not to line up perfectly with the ground truth images, but to maintain a crisp and likely representation of the world

We describe the two key components of the model briefly, namely, the multi-scale network architecture, as well as the combined loss function used to train the architecture.

4.2.1 Multi-scale Network

The multi-scale network, inspired by the reconstruction process of a Laplacian pyramid, is designed to mitigate the short-range dependencies of convolutions, which are limited due to the size of their kernels.

Let $Y = \{Y^1, ..., Y^n\}$ be a sequence of frames to predict from input frames $X = \{X^1, ..., X^m\}$. Let $s_1, s_2, ..., s_N$ be the sizes of the inputs of the network, where N denotes the number of scales we wish to train with. Let u_k be the upscaling operator that scales toward size s_k. Let X_k^i, Y_k^i denote the downscaled versions of X_i, Y_i of size s_k, and G_k' be a network that learns to predict $Y_k - u_k(Y_{k-1})$ from X_k and a coarse (low-level) guess of Y_k. We recursively define the network G_k that makes a prediction of $\hat{Y_k}$ of size s_k as follows.

$$\hat{Y_k} = G_k(X) = u_k(\hat{Y}_{k-1}) + G_k'(X_k, u_k(\hat{Y}_{k-1}))$$

Therefore, the network makes a series of predictions, starting from the lowest resolution, and uses the prediction of size s_k as a starting point to make the prediction of size s_{k+1}. At the lowest scale s_1, the network takes only X_1 as an input. The architecture is illustrated in Figure 4.

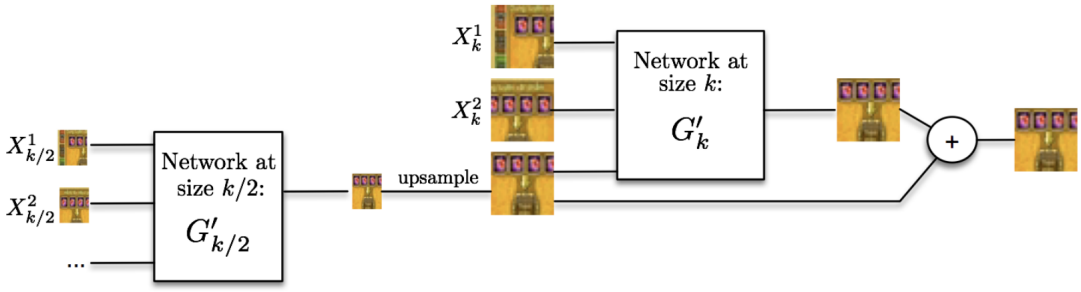

Figure 4: The multi-scale architecture proposed by Mathieu et al., which we adapted for our frame prediction experiments.

4.2.2 Combined Loss Function

The loss function for the multi-scale network is given as follows.

$$\mathcal{L}(X,Y) = \lambda_{adv}\mathcal{L}_{adv}(X,Y) + \lambda_{l_p}\mathcal{L}_{l_p}(X,Y) + \lambda_{gdl}\mathcal{L}_{gdl}(X,Y)$$

The different loss function components $\mathcal{L}_{adv}$ (adversarial loss), $\mathcal{L}_{l_p}$ (L-p loss), and $\mathcal{L}_{gdl}$ (image gradient loss) are weighted by $\lambda_{adv}, \lambda_{l_p}, \lambda_{gdl}$ respectively. Their definitions are provided below.

$$\mathcal{L}_p(X,Y) = l_p(G(X),Y) = ||G(X) - Y||_p^p$$

For $p = 1$, this is the *L-1 loss*. For $p = 2$, this is the *L-1 loss*. This loss component is the standard loss that minimizes the error between the ground truth frame and the predicted frame.

$$\begin{aligned}\mathcal{L}_{gdl}(X,Y) &= L_{gdl}(\hat{Y},Y) \\ &= \sum_{i,j} ||Y_{i,j} - Yi-1,j| - |\hat{Y}_{i,j} - \hat{Y}_{i-1,j}||^\alpha \\ &\quad + ||Y_{i,j-1} - Yi,j| - |\hat{Y}_{i,j-1} - \hat{Y}_{i,j}||^\alpha\end{aligned}$$

where $\alpha \geq 1$. The *image gradient loss* directly penalizes the differences of image gradient predictions. This loss component is relatively insensitive to low-frequency mismatches between prediction and target (e.g., adding a constant to all pixels does not affect the loss), and is more sensitive to high-frequency mismatches that are perceptually more significant (e.g. errors along the contours of an object). Thus, it serves to sharpen the image predictions.

$$\mathcal{L}_{adv}(X,Y) = \sum_{k=1}^{N}(L_b(D_k(X_k,Y_k),1) + L_b(D_k(X_k,G(X_k)),0))$$

where L_b denotes the *binary cross-entropy* loss, defined as:

$$L_b(Y,\hat{Y}) = -\sum_i \hat{Y}_i \log(\hat{Y}_i) + (1-\hat{Y}_i)\log(1-\hat{Y}_i)$$

where $Y_i \in \{0,1\}, \hat{Y}_i \in [0,1]$. The adversarial loss ensures that the predicted frames are as close in quality as possible to the ground truth frames, to the point where a fine-tuned discriminator network cannot distinguish between them.

5. Experimental Results

5.1. Approach I

Under this approach, different static images obtained from the representations generated in Section 3 are used to train two models: a variational autoencoder, and a Wasserstein Generative Adversarial Network (WGAN).

We trained a Wasserstein GAN [4], a variant of the original GAN formulation. To further improve the stability of the final model, we applied the gradient penalty method [8] instead of the weights-clipping method described in [4]. The WGAN loss function makes the training more stable since it is continuous over the generator's parameters (with mild assumptions). In addition to stability advantages, WGAN's loss function is designed to be correlated with the sample quality.

We used dense fully connected layers with ReLU units and hidden layer size of 64 for both the generator and the discriminator networks, which were trained using the Adam optimizer [10] with a learning rate of 1e-4 and mini-batch size of 256. The weights of both networks were initialized using the Xavier initialization method [6]. Due to the sheer amount of computation required to train a WGAN, the noise dimension was chosen to be 30, rather than the typically suggested size of 100.

The results obtained from training the GAN and running the generator network are presented in Figure 7. These results show what happens to the crowd density heatmaps as

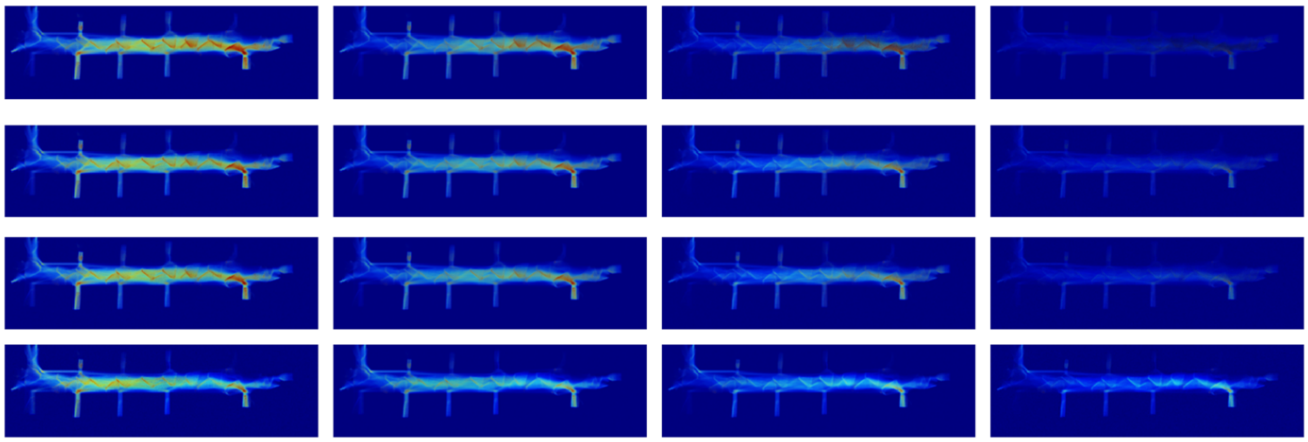

Figure 5: Parameter Change effect for VAE. By fixing one of the parameters and changing the other one, (moving alongside the horizontal/vertical axis) different images are generated. This can be interpreted as each dimension encoding some features underlying the data distribution.

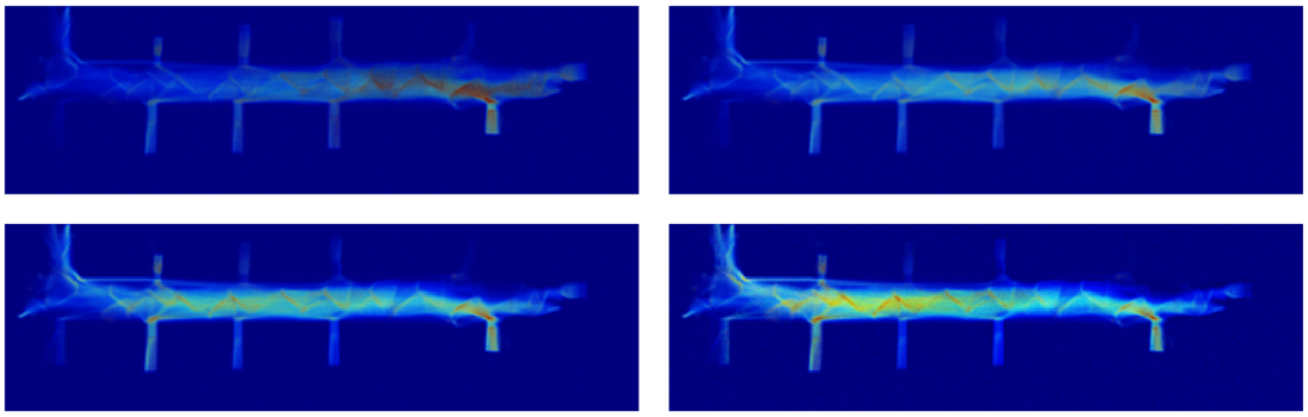

Figure 6: Interpolating in the latent space for VAE. The top right and bottom left images are randomly chosen points in the latent space and the other two pictures are the corresponding images obtained while interpolating across the latent space.

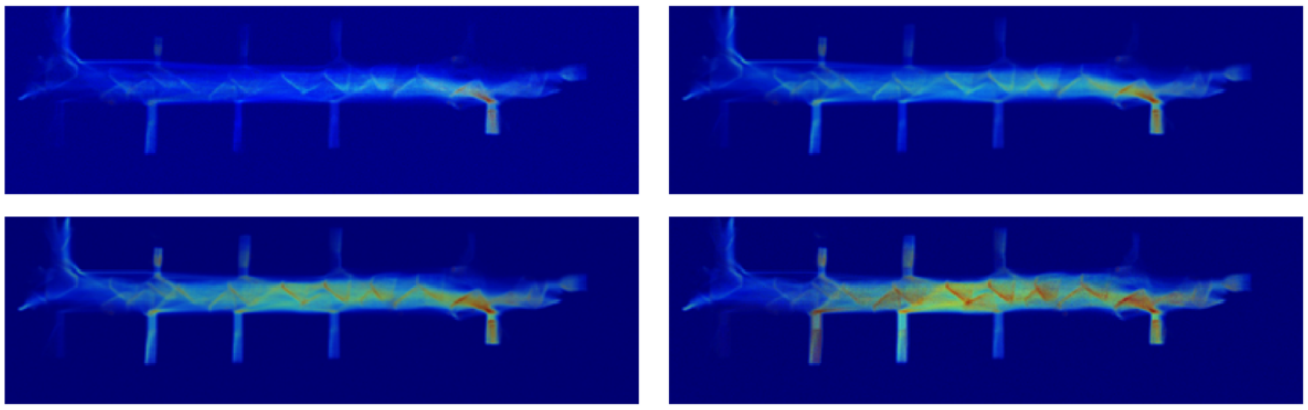

Figure 7: Interpolating in the latent space for WGAN. The top right and bottom left images are randomly chosen points in the latent space and the other two pictures are the corresponding images obtained while interpolating across the latent space.

we interpolate across the noise space. We can observe that the images change gradually. A reasonable interpretation of these images is that since the map gets more dense as we move across the noise space, this part of the noise space is somehow responsible for encoding the density and crowdedness of the hallways.

The architecture of the network used in VAE is also similar using fully connected layers with hidden layer dimension size 128. Mini batch and learning rate were chosen to be the same as before. For evaluating the VAE, we are interested in interpolating across the latent space so that we can visualize the different images generated for different values of the latent vectors and interpret the characteristic of the data distribution that each dimension of the latent vector represents. Thus, we chose the dimension of the latent vector to be 2 for our experiments.

The results of the VAE can be shown in Figures 5 and 6. Figure 5 shows how moving through one of the dimensions of the latent space while keeping the other one constant would change the generated images (The horizontal axis is one dimension and the vertical axis is the other dimension). We can observe from the images that the vertical axis is somehow associated with the density of the heatmap and even the characteristics of the map (such as bottleneck locations) while the change in the horizontal axis affects the crowdedness of the map.

Figure 6 also shows what effect interpolation in the latent space might have on the final generated images. Again, as can be seen by moving through the latent space of the VAE, images have a meaningful gradual change which is correlated with the heatmap density.

Another interesting point of note from our results is that in many of the more crowded images, the generated images have darker colors in some place close to the last room in the hallway. This implies that that region can become a bottleneck in times of crowdedness or panic.

5.2. Approach II

The architectures for the generator and discriminator networks that were used in our experiments are given below.

Scale	Feature Maps	Kernel Size
1	128,256,128	3,3,3,3
2	128,256,128	5,3,3,5
3	128, 256, 512, 256, 128	5,3,3,3,5
4	128, 256, 512, 256, 128	7,5,5,5,5,7

Table 1: Generator network

One significant advantage of the architecture outlined above is that since the discriminator has fully connected layers after applying the convolutions. Thus, the output of the last convolution layer must be flattened to connect the fully-connected layer. The size of this layer is dependent on the input size, and as a result blows up quickly if we are to train on the original image size (approx. 300×80). Pooling layers are not used in the current architecture as they may result in loss of resolution. Hence, we train on 32×32 patches of the original frames, that are sampled from a sequence of frames from our dataset. The obtained clips (a sequence of 4 32×32 frames and the corresponding ground truth output) are filtered to ensure that there is some movement present in them. This lets us train on a batch size of 8 clips. Since the generator is fully convolutional, during test time, input image of any size can be used.

Scale	Feature Maps	Kernel Size	FC
1	64	3	512,256
2	64,128,128	3,3,3	1024,512
3	128,256,256	5,5,5	1024,512
4	128,256,512,128	7,7,5,5	1024,512

Table 2: Discriminator network (FC = Fully connected layers; Kernels do not have padding applied)

We did not experiment with a lot of different learning rates for the discriminator and generator due to the sheer volume of computation involved. However, out of the different values we tried, we obtained the results presented in this report when we set the generator's learning rate to 0.00002 and the discriminator's learning rate to 0.02. The convolutions in the generator were padded and ReLU activation ($f(x) = \max(0, x)$) was applied to them, except for the last convolutional layer at each scale. At the last convolutional layer, tanh activation was applied to ensure that the output values lie in $[-1, 1]$. The convolutions in the discriminator are not padded and are fed directly into the fully connected layers with ReLU activations.

We trained and tested our architecture on both the agent-based representations and the density heatmap representations defined in Section 3. We trained the model on one day's worth of trajectory data, while another day's data was held out for testing the final generator model. For the heatmap representation, each frame represented the crowd density between a time interval of 1s, i.e. frames X^t and $X^{t+\delta}$ would correspond to the change in the heatmap for a duration of δ seconds ($\delta = 1$). Similarly, for the agent-based representation, the time lapse between two consecutive frames was 100ms. Each agent was represented by a 4×4 pixel square and was assigned a random color, that was retained across the duration of its trajectory.

Through our experiments, we aimed to answer the following questions:

- Can unsupervised deep models effectively build a sharp representation of the scene across multiple time steps?

- Can this representation be used as a predictive model to predict the movement of agents across time steps?

5.2.1 Qualitative Results

We first present a few qualitative results in the form of generated frames that are obtained by training our model on the sequential, agent-based representation in Figure 8. In Figure 9, we provide an example of the generated frame obtained by training our model on the heatmap representation. In both figures, the input frames X^1, X^2, X^3, X^4 are provided, as well as both Y^1 and $\hat{Y}^1$.

We also present another set of qualitative results using the same representations. In figures 10 and 11, we provide respectively, the ground truth frame $Y^1, ..., Y^8$, and the recursive predictions $\hat{Y}^1, ..., \hat{Y}^8$ that were generated using the following scheme: input frames X^1, X^2, X^3, X^4 were used to generate frame $\hat{Y}^1$, frames $X^2, X^3, X^4, \hat{Y}^1$ were used to generate frame $\hat{Y}^2$, and so on. The final predicted frame $\hat{Y}^8$ was generated purely from previously generated frames from our generated ($\hat{Y}^4, \hat{Y}^5, \hat{Y}^6, \hat{Y}^7$). The corresponding set of images for the heatmap representation is provided in figures 12 and 13 respectively.

5.2.2 Quantitative Results

For both the agent-based representations and the heatmap representations, we will report the average value of the following metrics across a held-out test set of clips.

Peak signal-to-noise ratio (PSNR) is defined as the ratio between the maximum possible power of a signal and the power of corrupting noise that affects the fidelity of its representation. It is used to evaluate the quality of the frame predictions. It can be defined for a given noise-free image Y and its noisy approximation $\hat{Y}$ (both of size M) as

$$PSNR(Y, \hat{Y}) = 10 \log_{10} \frac{\max_{\hat{Y}}^2}{\frac{1}{M} \sum_{i=0}^{N} (Y_i - \hat{Y}_i^2)}$$

where $\max_{\hat{Y}}$ is the maximum possible value of the image intensities.

To evaluate the relative sharpness between the ground truth frame and the generated frame, we use the *sharpness error* (SDE) based on the difference of gradients between Y and $\hat{Y}$ proposed in [15]:

$$\text{SDE} = 10 \log_{10} \frac{\max_{\hat{Y}}^2}{\frac{1}{N} (\sum_i \sum_j |(\nabla_i Y + \nabla_j Y) - (\nabla_i \hat{Y} - \nabla_j \hat{Y})|)}$$

where:

$$\nabla_i Y = |Y_{i,j} - Yi - 1, j|$$
$$\nabla_j Y = |Y_{i,j} - Yi, j - 1|$$

The PSNR and sharpness values for the agent-based representation are given in Table 3. The results are averaged over the entire test set, and are presented for the recursively predicted output frames $\hat{Y}^i, i \in 1, 2, 3, 4$.

Predicted frame $\hat{Y}^i$	PSNR	Sharp.diff.
$i = 1$	35.75	24.75
$i = 2$	33.40	23.57
$i = 3$	31.79	22.72
$i = 4$	30.66	22.09

Table 3: Average PSNR and sharpness values for the first 4 predicted frames in the test set for the agent-based representation

The PSNR and sharpness values for the heatmap-based representation are given in Table 4. The results are averaged over the entire test set, and are presented for the recursively predicted output frames $\hat{Y}^i, i \in 1, 2, 3, 4$.

Predicted frame $\hat{Y}^i$	PSNR	Sharp.diff.
$i = 1$	35.62	26.93
$i = 2$	31.64	24.98
$i = 3$	29.88	23.95
$i = 4$	29.08	23.34

Table 4: Average PSNR and sharpness values for the first 4 predicted frames in the test set for the heatmap-based representation

As we can observe from the results on both the representations, the images obtained are sharp and indicative of the ground truth image, though the quality of the images deteriorates as the prediction process becomes less reliant on the ground truth images and more reliant on previously generated frames. The agent-based representation fairs slightly better than the heatmap representation. This could be attributed to the fact that the heatmap representation is inherently suppressing the information about the movement of individual agents across time, whereas the agent-based representation is able to use the existence of this information to its advantage and generate sharper images.

6. Conclusion

Approach I shows that by using unsupervised deep models such as VAEs and GANs, insightful information can be extracted from the provided representations of crowd trajectory data, and encoded in a more concise manner, perhaps as input to another machine learning model.

From our results in Approach II, we are able to conclude that unsupervised deep models can not only effectively build a sharp representation of the scene across multiple time steps, they can also effectively use this representa-

Figure 8: In the top row, we provide the sequence of four frames that are presented as an input to the generator during test time. In the bottom row, the image on the **left** is the ground truth frame. The image on the **right** is the frame generated by our generator. These figures are also included as part of our submission.

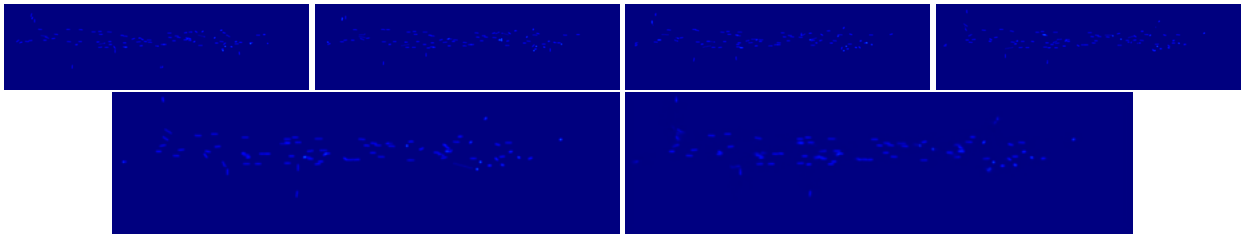

Figure 9: In the top row, we provide the sequence of four frames that are presented as an input to the generator during test time. In the bottom row, the image on the **left** is the ground truth frame. The image on the **right** is the frame generated by our generator. These figures are also included as part of our submission.

Figure 10: Ground truth frames $Y^1, ...Y^8$ for the agent-based representation. These figures are also included as part of our submission.

Figure 11: Recursively generated frames $\hat{Y}^1, ..., \hat{Y}^8$ for the agent-based representation. These figures are also included as part of our submission.

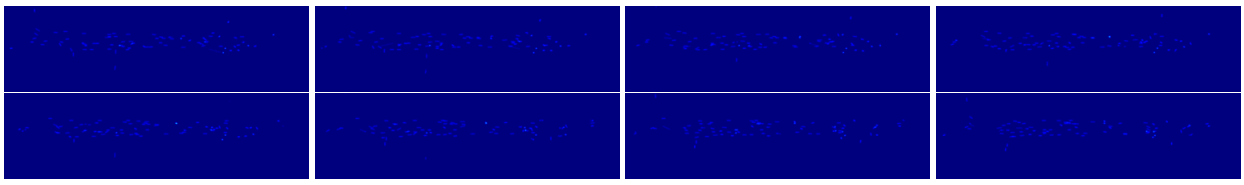

Figure 12: Ground truth frames $Y^1, ...Y^8$ for the heatmap representation. These figures are also included as part of our submission.

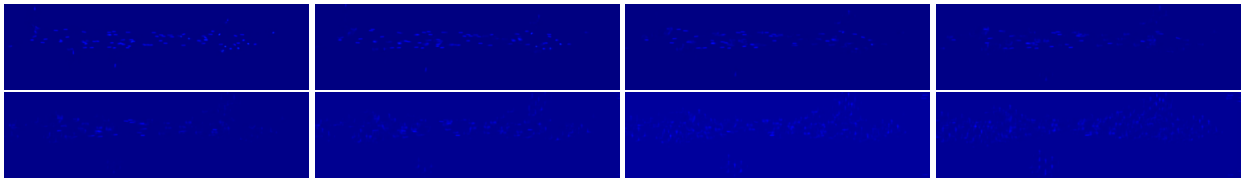

Figure 13: Recursively generated frames $\hat{Y}^1, ..., \hat{Y}^8$ for the heatmap representation. These figures are also included as part of our submission.

tion to build a predictive model of the movement of agents across time steps.

We also experimented with other GAN models such as conditional GANs [16] and InfoGANS [5]. Under this approach, the density of the image could have been used as the discrete latent variable to the GAN, so that it could be trained to generate images corresponding to a particular label. However, we did not meet with much success in training these models.

It would have been interesting to use the low-dimensional representations constructed using Approach I as an input for the predictive model as Approach II. Essentially, Approach I would have focused on building a representation of the spatial semantic information present in the image, and Approach II would have focused on building a representation of the temporal information that is present across a sequence of frames.

Contributions

Kshitij worked on constructing the representations of the data from the raw trajectories. Sepehr & Kshitij collaborated on Approach I. Shardul & Aravind collaborated on Approach II. Shardul & Sepehr consolidated the experiments and generated and collected the figures for the report. Aravind & Kshitij collaborated on the writing of the report.

Acknowledgements

All the architectures described in this report were implemented using Tensorflow [1]. We would like to thank Alexandre Alahi for providing access to the crowd dataset used in this report. The implementation of WGAN and VAE for Approach I are loosely based on Agustinus Kristiadi's implementation. The implementation of Approach II, as well as the frame pre-processing was adapted from Matthew Cooper's implementation. We would like to acknowledge the assistance provided to us by Google Compute Engine for training these sophisticated models. All supplemental code and figures have been submitted along with this report.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 10

[2] A. Alahi, V. Ramanathan, and L. Fei-Fei. Socially-aware large-scale crowd forecasting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2203–2210, 2014. 1, 2

[3] S. Ali and M. Shah. Floor fields for tracking in high density crowd scenes. *Computer Vision–ECCV 2008*, pages 1–14, 2008. 1

[4] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 5

[5] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2172–2180, 2016. 10

[6] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. 5

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014. 1

[8] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville. Improved training of wasserstein gans. *arXiv preprint arXiv:1704.00028*, 2017. 5

[9] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006. 4

[10] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[11] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1, 4

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[13] T. Li, H. Chang, M. Wang, B. Ni, R. Hong, and S. Yan. Crowded scene analysis: A survey. *IEEE transactions on*

circuits and systems for video technology, 25(3):367–386, 2015. 1

[14] Y. Lv, Y. Duan, W. Kang, Z. Li, and F.-Y. Wang. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865– 873, 2015. 1

[15] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015. 4, 8

[16] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014. 10

[17] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning*, pages 843–852, 2015. 4

[18] C. Zhang, H. Li, X. Wang, and X. Yang. Cross-scene crowd counting via deep convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015. 1

[19] B. Zhou, X. Wang, and X. Tang. Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2871– 2878. IEEE, 2012. 1

[20] W. Zhu, Y. Sun, and Y. Sheng. Wave-dynamics simulation using deep neural networks. 2