Diffusion model uniform manifold filtering for classification of small datasets with underrepresented classes: Application to chromosomal aberration microscopy detection

Quentin Tallon, Juan Martinez, Eric Gregoire, Pascale Fernandez, Delphine Dugue, Gaetan Gruel, Charles Kervrann, Mohamedamine Benadjaoud

Diffusion model uniform manifold filtering for classification of small datasets with underrepresented classes: Application to chromosomal aberration microscopy detection

Quentin Tallon[a, b], Juan Martinez[a], Eric Gregoire[a], Pascale Fernandez[a], Delphine Dugue[a], Gaetan Gruel[a], Charles Kervrann[b], and Mohamedamine Benadjaoud[a]

[a]Accidental Exposure Radiobiology Laboratory, Institute for Radioprotection and Nuclear Safety (IRSN), Fontenay-aux-Roses, France
[b]SAIRPICO Project-Team, National Institute for Research in Digital Science and Technology (Inria), Rennes Cedex, France

A frequent problem in biomedical machine learning is the issue of imbalanced classes, especially when datasets are small, which restricts the performance of deep learning methods. To address this issue, generative models are often used to generate additional synthetic data. Specifically, image-to-image models can transform input images to match the characteristics of target images. However, training such models on small datasets can affect the quality of synthetic samples. We propose a new method to filter generative outputs of an image-to-image Brownian Bridge Diffusion Models (BBDMs) using Uniform Manifold Approximation and Projection (UMAP) dimension reduction of a real data classifier's feature space. We apply this methodology to filter synthetic chromosomal aberrations generated from the blue DAPI-colored channels in the context of cytogenetic Fluorescence In Situ Hybridization (FISH) microscopy. Our method shows that such filtered synthetic data significantly enhance classification performance compared to the state of the art CycleGAN and could potentially be applied to a variety of other generative models.

Keywords: Biomedical imaging, Generative models, Image-to-image translation, Filtering, Classification, Chromosomal aberrations

1. Introduction

Biomedical imaging involves the acquisition, processing, and visualization of structural or functional images of living organisms or systems. The complexity of biomedical data often necessitates experts to annotate the data, making the annotation process both time-consuming and expensive [1]. Consequently, most biomedical datasets are limited in size, which presents significant challenges for machine learning tasks [2–4].

For a classification task, the elements of interest, such as specific biological anomalies, are often rare [5, 6]. This rarity is particularly problematic to achieve high classification performance in unbalanced and small datasets. Several techniques have been developed, such as re-sampling and cost-sensitive learning [7–11]. Re-sampling [7, 10, 11] balances class distribution by over-sampling or under-sampling, while cost-sensitive learning [8,9] assigns different misclassification costs to classes, both requiring careful implementation to avoid over-fitting, under-fitting, or high false positive rates. These methods are often inefficient in the context of small biological datasets [2–4]. Moreover, it has been found that for image classification in deep learning, underrepresented classes can benefit from features learned from overrepresented classes. This arises from the observation that the first layers of a convolutional network encode low level, generic features, while deeper layers progressively encode high-level, class-specific features [12]. As such, all classes benefit from good low-level features. Therefore, if one seeks to detect a rare object class with deep learning, it is useful to use overrepresented classes.

In this study, we address the issue of binary classification, where one class is significantly underrepresented. Generating synthetic data provides an opportunity to balance the dataset by producing additional samples of the underrepresented class [5, 6]. Diffusion models [13, 14] have been shown to produce better quality samples than Generative Adversarial Networks [15–17] (GANs), which is particularly important for the complex biomedical data used in our classification task [18]. On small datasets, GANs often suffer from discriminator overfit, leading to mode collapse. Image-to-image [19–22] diffusion models, such as Brownian Bridge Diffusion Models (BBDMs) [22, 23] hence offer an attractive alternative,

but the quality of their synthetic data is still compromised when trained on small datasets [24–26]. In this case, there is a need for a robust filtering mechanism to select the most realistic samples. We propose a novel approach to filter synthetic samples using the feature space of a classifier [27] trained on real data. By applying Uniform Manifold Approximation and Projection (UMAP) [28] dimension reduction on the penultimate layer, we can map the feature space of real data and identify the region corresponding to the underrepresented class. Generated samples are then projected into this feature space, and only those that are closest to the real data underrepresented class are retained.

To demonstrate the efficiency of our approach, we investigated the classification of chromosomes in Fluorescence In Situ Hybridization (FISH) [29,30] microscopy images where the occurrence of chromosomal aberrations is rare. More precisely, the artificial chromosomal translocations generated from the blue DAPI (4',6-diamidino-2-phenylindole) channels of unannotated chromosome images were filtered through the UMAP [28] feature space to provide additional translocation samples to the real dataset. Our results indicate that the proposed method can significantly enhance classification performance based on diffusion models compared to other generative strategies (CycleGAN) and provide a viable solution to improve classification scores in small and imbalanced biomedical datasets.

2. Method

2.1 Brownian Bridge Diffusion Model (BBDM)

The Brownian Bridge Diffusion Model (BBDM) [22] offers a novel approach to image-to-image translation by modeling the process as a stochastic Brownian bridge. This method differs from traditional Denoising Diffusion Probabilistic Models (DDPMs) [13] by directly mapping transformations between two image domains through a bidirectional diffusion process, rather than using a conditional generation process. This approach mitigates the gap between distinct domains and enhances the stability and theoretical grounding of image synthesis.

2.1.1 Forward and Backward Processes

In the forward diffusion process, the BBDM starts with clean data x_0 sampled from the data distribution $q_{\text{data}}(x_0)$ and gradually adds noise to transition towards a standard Gaussian distribution. The key equations governing this process are:

$$q_{BB}(x_t|x_0, y) = \mathcal{N}(x_t; (1 - m_t)x_0 + m_t y, \delta_t \mathbf{I}), \tag{1}$$

where x_0 is the initial state, x_t is the intermediate state, y is the target image, $m_t = \frac{t}{T}$ is the interpolation coefficient with T the number of diffusion steps, $\mathbf{I}$ as identity, and δ_t represents the variance schedule. The forward process in the BBDM can thus be seen as a Markov chain with transition probabilities defined as:

$$q_{BB}(x_t|x_{t-1}, y) = \mathcal{N}(x_t; \frac{1 - m_t}{1 - m_{t-1}} x_{t-1} + \left(m_t - \frac{1 - m_t}{1 - m_{t-1}} m_{t-1} \right) y, \delta_{t|t-1} \mathbf{I}), \tag{2}$$

where $\delta_{t|t-1} = \delta_t - \frac{\delta_{t-1}(1-m_t)^2}{(1-m_{t-1})^2}$. This process ensures a smooth transition from the initial state to the target state by conditioning the intermediate states on both endpoints, forming a bridge.

The reverse diffusion process aims to predict the previous state x_{t-1} from the current state x_t. Unlike traditional diffusion models, which start from pure noise, BBDM begins from the target image y. The reverse transition probability is defined as:

$$p_\theta(x_{t-1}|x_t, y) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \tilde{\delta}_t \mathbf{I}), \tag{3}$$

where $\mu_\theta(x_t, t)$ is the predicted mean and $\tilde{\delta}_t$ is the fixed variance. The mean $\mu_\theta(x_t, t)$ is predicted using a neural network parameterized by θ. This network is trained to minimize the difference between the predicted noise and the actual noise added in the forward process.

2.1.2 Training Objective

The training objective of the BBDM is to optimize the Evidence Lower Bound (ELBO), formulated as:

$$\text{ELBO} = -\mathbb{E}_q \left[\text{KL}(q_{BB}(x_T|x_0, y) \| p(x_T|y)) + \sum_{t=2}^{T} \text{KL}(q_{BB}(x_{t-1}|x_t, x_0, y) \| p_\theta(x_{t-1}|x_t, y)) - \log p_\theta(x_0|x_1, y) \right]. \quad (4)$$

Since $x_T = y$ in the Brownian Bridge process, the first term becomes a constant and can be ignored. The training focuses on minimizing the Kullback-Leibler (KL) divergence between the forward and reverse processes at each step, ensuring that the model learns to generate images that closely match the target distribution.

To improve efficiency, the BBDM employs an accelerated sampling process similar to the Denoising Diffusion Implicit Models (DDIM) [31]. This involves using a non-Markovian process that maintains the same marginal distributions as the Markovian inference process. By selecting a subset of relevant variables and optimizing the transition probabilities, the sampling process can be significantly sped up without compromising the quality of the generated images.

The BBDM framework thus provides a robust method for image-to-image translation, leveraging the properties of the Brownian Bridge to achieve high fidelity and diverse synthetic image generation.

2.2 ResNet Classifier for Feature Extraction

Residual Networks (ResNets) [27] provide a robust framework for deep learning by mitigating the vanishing gradient problem through the use of residual connections. These connections allow the network to learn residual functions with reference to the layer inputs, which simplifies the optimization of deeper networks.

The architecture of ResNet is designed to enable the training of very deep networks. The key innovation in ResNet is the introduction of residual learning blocks. A basic residual block can be expressed as:

$$\mathbf{y} = \mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x}) + \mathbf{x}, \quad (5)$$

where σ denotes the ReLU activation function, and $\mathbf{W}_1$ and $\mathbf{W}_2$ are the weights of the convolutional layers.

For deeper networks, a bottleneck design is often used to reduce the number of parameters and computational complexity. A bottleneck residual block is defined as:

$$\mathbf{y} = \mathbf{W}_3\sigma(\mathbf{W}_2\sigma(\mathbf{W}_1\mathbf{x})) + \mathbf{x}, \quad (6)$$

where $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$ are the weights of the 1x1, 3x3, and 1x1 convolutional layers, respectively.

The trained ResNet model provides a robust feature extractor [12] based on the penultimate layer (layer just before the last fully connected layer) generating high-dimensional feature vectors that are crucial for subsequent steps in our methodology.

2.3 Uniform Manifold Approximation and Projection (UMAP)

Uniform Manifold Approximation and Projection (UMAP) [28] is a manifold learning technique for dimension reduction that is highly effective for both visualization and general-purpose machine learning tasks. UMAP builds on mathematical foundations in Riemannian geometry and algebraic topology, creating a practical, scalable algorithm applicable to real-world data.

The UMAP algorithm approximates the manifold on which the data lies by assuming that the data is uniformly distributed. This assumption, while simplifying, is beneficial for theoretical reasons similar to those employed in Laplacian Eigenmaps.

The core of UMAP's methodology is constructing a topological representation of the data through fuzzy simplicial sets. The steps involved in this process are:

1. Nearest Neighbor Search: UMAP starts by finding the nearest neighbors for each data point in the high-dimensional space. This step is crucial for constructing the local fuzzy simplicial set memberships.

2. Constructing High-dimensional Fuzzy Simplicial Sets: Local fuzzy simplicial set memberships are defined for each data point based on the distances to its nearest neighbors. The memberships are calculated as:

$$v_{j|i} = \exp\left(\frac{-d(x_i, x_j) - \rho_i}{\sigma_i}\right), \tag{7}$$

where $d(x_i, x_j)$ is the distance between data points x_i and x_j, ρ_i is the distance to the nearest neighbor of x_i, and σ_i is a normalization factor determined heuristically.

3. Symmetrizing the Memberships: The local memberships are symmetrized using a fuzzy set union to create a topological structure:

$$v_{ij} = v_{j|i} + v_{i|j} - v_{j|i}v_{i|j}. \tag{8}$$

4. Optimizing the Low-dimensional Representation: The low-dimensional similarities are given by a smooth, differentiable function of the Euclidean distance between points:

$$w_{ij} = \left(1 + a\|y_i - y_j\|^{2b}\right)^{-1}. \tag{9}$$

Here, a and b are hyperparameters that control the tightness and balance of the embedding. The optimization process aims to minimize the cross-entropy between the high-dimensional and low-dimensional fuzzy simplicial sets, ensuring that the local structure of the data is preserved in the embedding.

2.4 Feature Space UMAP Filter

2.4.1 Feature Extraction and Visualisation

The extracted feature vectors from the ResNet classifier represent the high-dimensional feature space of the real data. These feature vectors serve as the input for the UMAP algorithm, which will project them into a lower-dimensional space. Using the low-dimensional UMAP projection, we identify the region corresponding to the underrepresented class using gaussian mixture density estimation. This involves examining the UMAP projection to localize the underrepresented data class in the feature space. A filter is created based on the confidence ellipsoid contour of the gaussian distribution component describing the most adequately the underrepresented class, in order to retain only those synthetic samples that are close to the real data samples of the underrepresented class. The confidence level of the gaussian component contour was defined according to its weight in the mixture model.

2.4.2 Filtering Synthetic Samples

The generated synthetic samples are projected into the UMAP space. Using the filter, we select only the samples that fall within the identified region, ensuring that the synthetic data closely resembles the real data of the underrepresented class. The pipeline is shown in Figure 1.

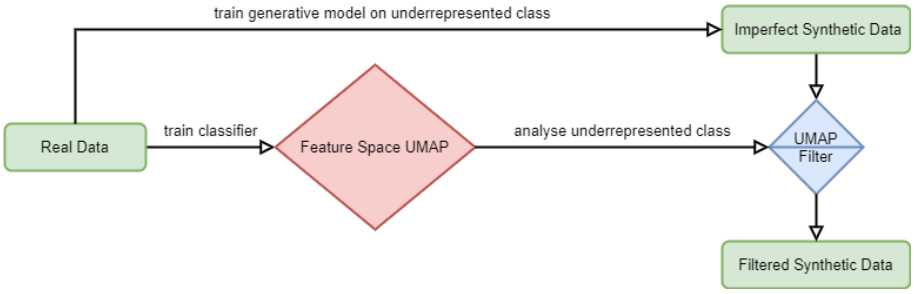

Figure 1. The Proposed UMAP Filter Pipeline. Real data is used for the training of the classifier and the generative models. The feature space UMAP projection is exploited to filter synthetic data.

3. Experiments

3.1 Dataset

Following exposure to ionizing radiation, it is necessary to refine the assessment of the dose received (sorting of victims, radiotherapy control, etc.). Among the available techniques, biological dosimetry based on cytogenetic imaging consists of counting chromosomal aberrations (CA) within circulating lymphocytes. These aberrations can be unstable or stable, the latter being more suitable to perform retrospective dosimetry years after exposure. The main stable aberration is the chromosomal translocation which persists in cells during cell division and can serve as a basis for a dosimetric reconstruction several years after exposure. They can be identified through color colocalizations in Fluorescence In Situ Hybridization (FISH) imaging [6, 29]. The CA manual scoring procedure is long and tedious, requiring trained biologists, and to our knowledge no automated solutions are available. The images are acquired by fluorescent staining of 3 pairs of chromosomes (painting of chromosome 2 in red, chromosome 4 in green and chromosome 12 in yellow). The blue channel contains a representation of the DAPI dye which sticks to all genetic material. The red and green channels contain representations of the fluorescent painting.

In this study, we define 2 chromosome classes. Class 1 (Normal), chromosomes with uniform fluorescent staining. They appear as red, green or yellow in the merged RGB images. Class 2 (Aberration), chromosome aberrations (underrepresented), specifically chromosomal translocations.

Additionally, we utilized Cellpose [1, 32] to segment the chromosomes and construct the instance database for the generative model training. Cellpose is a generalist, deep learning-based segmentation method designed to precisely segment cells from a wide range of image types without requiring model retraining or parameter adjustments. The core of Cellpose's methodology is a neural network that predicts spatial gradients based on a U-Net [33] architecture. This network down samples convolutional maps before up sampling them in a mirror-symmetric fashion, facilitating accurate segmentation by grouping pixels that converge to the same point. This segmentation process is crucial for focusing the analysis on individual chromosomes and their aberrations, thereby enhancing the accuracy of our generative models and subsequent classifications. Examples of segmented chromosome aberrations are shown in Figure 2.

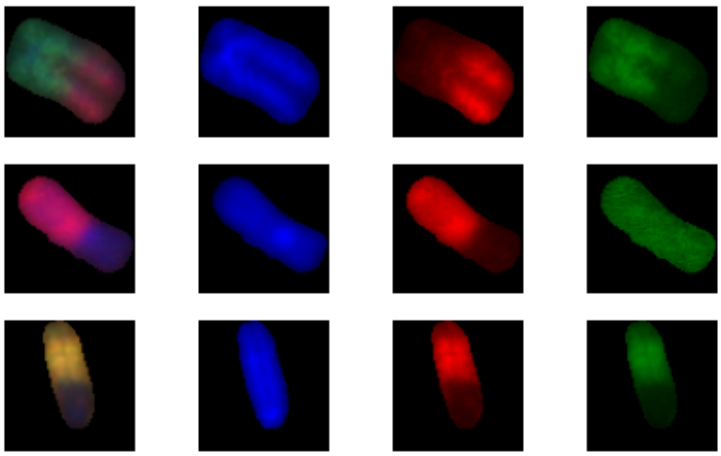

Figure 2. Example of Segmented Translocated Chromosomes. First column is the RGB image showing typical color co-localization present in translocations. Second, third and fourth columns are blue (DAPI), red and green channels, respectively.

Table 1 shows the distribution of annotated data used for training, validation, and testing, which highlights the imbalance between normal and aberrant chromosomes, a critical aspect to address in our analysis. Table 2 describes two datasets of unannotated metaphases and chromosomes. These unannotated datasets are used for inferring synthetic data when the generative model is trained in order to augment the classification task.

Table 1. Real Annotated Dataset used to train the classifier and the generative models. The aberration class in bold is strongly underrepresented.

	Train	Val	Test
Normal: fully colored chromosomes	10849	4095	3114
Aberration: chromosomal translocations	**1021**	**397**	**263**

Table 2. DAPI Chromosome Dataset used to infer the image-to-image models. Chromosomes are extracted from images of lymphocytes frozen in the metaphase stage of cellular division.

	Metaphases	Chromosomes
DAPI Data	9130	31507

3.2 Classification Baseline

To establish a baseline for our classification task, we trained a Residual Network [27] model on the real dataset of Table 1. ResNet, known for its robustness and accuracy in image classification tasks, serves as an appropriate choice for distinguishing between normal chromosomes and chromosomal aberrations in our FISH images. The training process involved optimizing the network to minimize classification errors using the annotated dataset described in the previous section.

The impact of data augmentation on model performance was a key focus of our experiments. We aimed to compare the classification scores with and without the use of data augmentation to evaluate its effectiveness on future synthetic data. In our study, we applied simple horizontal and vertical flip as well as rotation augmentations to the training images.

The results of the baseline classification, with and without data augmentation, are summarized in Table 3 . Details on classification results can be found in appendix B.

Table 3. Baseline ResNet Classification Scores on Real Data for the Underrepresented Aberration Class.

	Precision	Recall	F1-Score
Without Augmentation	0.81 ± 0.04	0.72 ± 0.03	0.76 ± 0.02
With Augmentation	0.83 ± 0.02	0.78 ± 0.01	0.81 ± 0.01

3.3 Generative Model Training and Sampling

The BBDM [22] was trained for an image-to-image task on the underrepresented class of the annotated dataset to generate synthetic images of chromosomal aberrations from blue DAPI channel inputs of a chromosome images. The training process involved configuring the BBDM with parameters and hyperparameters closely aligned with those used in the original paper. During training, the model parameters such as the number of diffusion and sampling steps were carefully tuned to achieve the best possible results. The training loss was monitored to ensure that the model converged to a state where it could generate high-fidelity synthetic samples.

However, the quality of the generated samples varied, with some samples closely resembling real chromosomal aberrations while others did not. This variation in quality can be seen in Figure 3. In the first row of Figure 3, we present a bad sample: the first image shows the original chromosome, the second image shows the blue DAPI-colored channel, and the third image shows the BBDM-generated sample. The BBDM-generated sample in the first row does not adequately resemble a chromosomal aberration. Conversely, the second row of Figure 3 illustrates a good sample: the generated image closely matches the characteristics of chromosomal aberrations. This demonstrates the potential of the BBDM when the training conditions and parameters are optimal. More examples of generated samples can be found in the appendix A.

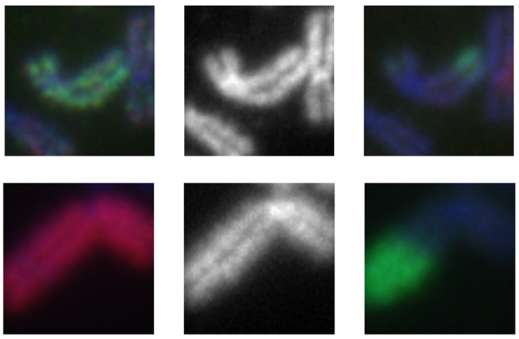

Figure 3. Example of BBDM Synthetic Chromosomes inference. First column is the normal chromosome, second is the blue channel input (DAPI), third is the synthetic aberration output.

Additionally, we tested the CycleGAN [20] model for generating synthetic chromosomal aberrations. The results from CycleGAN showed evidence of mode collapse, attributed to discriminator overfit, which is a common issue in GANs when dealing with small datasets. This is illustrated in Figure 4 and highlights the advantage of diffusion models over GANs in producing high-quality samples.

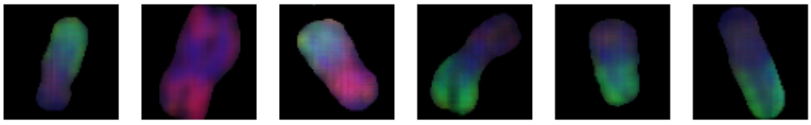

Figure 4. Example of CycleGAN Synthetic Chromosomes. The color colocalizations are often unsatisfactory and most chromosomes are colored in green, indicating mode collapse.

To further evaluate the quality of the synthetic samples, we conducted an Structural Similarity Index Measure (SSIM) analysis. The process involved calculating the SSIM matrix between training samples and synthetic samples and examining the maximum SSIM score for each synthetic sample. We then analyzed the density estimations of these maximum SSIM scores. This analysis shown in Figure 5 provided additional insights into the similarity and quality of the synthetic samples compared to the real training data, and it showed that BBDM outperformed CycleGAN in generating high-quality synthetic samples.

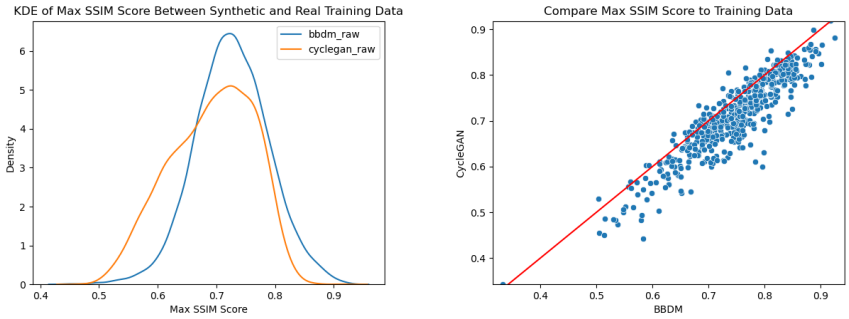

Figure 5. Structural Similarity Index Measure (SSIM) Analysis comparing BBDM and CycleGAN inference quality. Left is the density estimation of the max SSIM score between synthetic and real data. Right is the associated maximum SSIM scatter plot. Both representations indicate that the BBDM's scores are mostly greater.

3.4 Feature Space UMAP Filter

In this section, we apply the UMAP [28] methodology presented in section 2.4 to our real and synthetic datasets. The primary goal is to visualize and analyze the feature space of the chromosomal data to understand the quality and distribution of the generated synthetic samples.

We start by extracting feature vectors from the real and synthetic images using the ResNet model trained on the real dataset as described earlier. These feature vectors capture the essential characteristics of the chromosomes and serve as input for the UMAP algorithm. The UMAP algorithm is then applied to reduce the dimensionality of feature vectors of the real data (translocations and colored chromosomes), enabling us to visualize the real data in a lower-dimensional space. The feature vectors of the synthetic translocation chromosomes were then projected in this UMAP embedding space to visualize their proximity with the real translocations.

Figure 6 shows the UMAP projection of the real data feature space. This figure includes two plots: one showing all the data (both normal chromosomes and aberrations) and another showing only the chromosomal translocations. The UMAP visualization helps in understanding the distribution and clustering of the chromosomal aberrations within the feature space.

Next, we project the feature vectors of the synthetic data into the same UMAP feature space. Figure 6 illustrates the projection of the synthetic dataset generated by the BBDM. By comparing these projections with the real data, we can assess the quality of the synthetic samples through their proximity to the real chromosomal translocations.

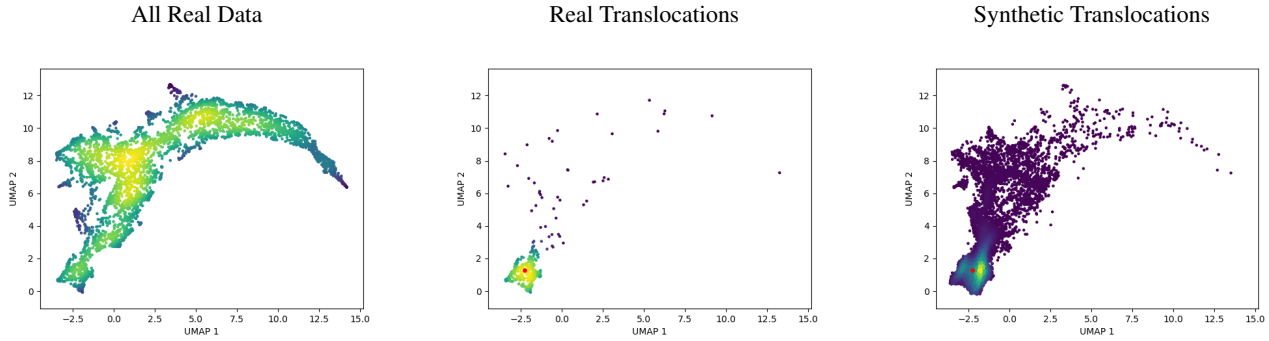

Figure 6. UMAP Space Projections and their probability density estimation. Left is all Real Data classes, middle is Real Data underrepresented translocation class, right is BBDM Synthetic translocation Data.

As expected, the UMAP projections reveal that the distribution of the synthetic data is different from the real aberrations. While some synthetic samples were projected closely to the real chromosomal aberration subspace, others do not. With the UMAP visualizations these disparities become evident, where the synthetic data points are more dispersed compared to the tighter clustering of the real aberrations. This dispersion explains why some synthetic samples are considered bad, as they do not adequately capture the localized patterns observed in real chromosomal aberrations.

The distribution of the UMAP coordinates of the translocated chromosomes can be captured by a two components gaussian mixture model. The gaussian component with the highest weight adequately covers the transclocation UMAP region. To improve the quality of the synthetic data, we filter the synthetic samples by the ellipsoide contour of this gaussian component at a given confidence level. This filter can be tuned depending on the classification results, allowing for dynamic adjustment of the synthetic data quality based on the model's performance. Table 4 shows the number of chromosomes before and after applying the UMAP filter for two datasets.

Table 4. BBDM Synthetic Data Before and After UMAP Filter.

Dataset	Before Filter	After Filter
BBDM Data	31507	24619

By identifying and understanding the differences in the distribution of real and synthetic data, the UMAP analysis underscores the importance of the feature space localization for retaining high-quality synthetic samples.

3.5 Classification Evaluation

The effectiveness of our approach was evaluated by comparing the classification performance of ResNet models trained on the real dataset with and without the inclusion of synthetic data. The results demonstrate the impact of our synthetic data generation and filtering methods on improving the classification of chromosomal translocation.

As established in section 3.2, the baseline ResNet model, trained on augmented real data, achieved scores seen in Table 3. These results highlight the benefit of data augmentation in enhancing the model's performance. However, to further address the class imbalance in the dataset and improve classification accuracy, we incorporated synthetic data generated using the BBDM.

Table 5 summarizes the classification scores when the model was trained without any additional augmentation techniques, using real and synthetic data. The results show that the inclusion of synthetic data generated from the DAPI data improved the recall rates compared to the model trained on real data alone. This indicates that the synthetic data effectively increased the model's ability to identify true chromosomal aberrations, although with a slight trade-off in precision.

We also tested the inclusion of synthetic data generated by CycleGAN and compared it with BBDM-generated data. Table 5 includes the classification scores with CycleGAN-generated data, which further demonstrates the superiority of BBDM in this context. CycleGAN's high precision and low recall provides additional insight of mode-collapse.

Table 5. Classification Scores Without conventional Data Augmentation.

	Precision	Recall	F1-Score
Real Data	0.81 ± 0.04	0.72 ± 0.03	0.76 ± 0.02
Real Data + BBDM Data	$\mathbf{0.75 \pm 0.07}$	$\mathbf{0.84 \pm 0.02}$	$\mathbf{0.79 \pm 0.04}$
Real Data + CycleGAN Data	0.85 ± 0.03	0.14 ± 0.09	0.23 ± 0.05

Table 6 presents the classification scores when data augmentation was applied in addition to incorporating the UMAP filtered synthetic data. The inclusion of synthetic data along with data augmentation resulted in further improvements in the classification performance. The pooled data (real and synthetic data) achieved higher F1 scores compared to the baseline model trained on real data with augmentation. Specifically, the model achieved an F1 score of 84%, the highest among all configurations, with no tradeoff in precision.

Table 6. Classification Scores With conventional Data Augmentation on Real Data.

	Precision	Recall	F1-Score
Augmented Real Data	0.83 ± 0.02	0.78 ± 0.01	0.81 ± 0.01
Augmented Real Data + BBDM Data	$\mathbf{0.84 \pm 0.02}$	$\mathbf{0.83 \pm 0.02}$	$\mathbf{0.84 \pm 0.01}$

These results demonstrate that our synthetic data generation and filtering methods significantly enhance the model's performance in classifying chromosomal aberrations. The use of UMAP to filter and retain the most qualitative synthetic samples ensures that the synthetic dataset closely resembles the real data distribution, leading to more accurate and reliable classification results.

4. Discussion

In this work, we tackled the challenge of classifying chromosomal aberrations in FISH [6, 29] images by utilizing synthetic data generation and advanced filtering techniques. We began by simplifying and preprocessing our dataset, eliminating outliers, and segmenting chromosomes using Cellpose [1, 32] to ensure clean and reliable data for training. The baseline ResNet [27] model, trained on this cleaned dataset, demonstrated robust performance, which was further enhanced through conventional data augmentation techniques.

Given the limitations posed by the size of the dataset and the rarity of chromosomal aberrations, we employed the BBDM [22] to generate new synthetic images. This approach allowed us to augment the dataset significantly, addressing the class imbalance issue. To evaluate the effectiveness of the BBDM model, we compared the results with those obtained using CycleGAN [20]. The CycleGAN model showed issues with discriminator overfitting and mode collapse, resulting in lower quality samples.

To refine the quality of the synthetic data, we applied UMAP [28] for feature space analysis and introduced a filtering mechanism. By creating a filter based on a fitted gaussian mixture distribution of the 2D real data UMAP coordinates, we retained only the most realistic translocations. This filtering process significantly improved the quality of the synthetic dataset.

Our experiments demonstrated that the inclusion of synthetic data, both with and without additional conventional data augmentation, resulted in substantial improvements in the model's performance. The combination of synthetic data and conventional data augmentation yielded the highest F1 scores, indicating a more balanced and accurate classification. In scenarios where acquiring sufficient real data is challenging or expensive, our methodology presents a viable solution to enhance model performance through synthetic data generation and rigorous filtering. Future work could focus on the application of our UMAP filtering to other generative models and extending the application of this methodology to other types of biomedical imaging data.

In summary, this study underscores the potential of combining generative models and advanced feature space analysis to address data imbalance in biomedical image classification tasks. Our integrated framework of synthetic data generation, UMAP filtering, and data augmentation provides a robust solution for improving classification accuracy in challenging scenarios.

Acknowledgments. This work was supported by ANR (AID/DGA, ASTRID) INCREASED Project, Grant Number ANR-20-ASTR-0005-01. Computing was performed on the Inria Rennes computing grid facilities partly funded by France-BioImaging Infrastructure (French National Research Agency ANR-10-INBS-04-07, Investments for the Future). A special thanks to Mrs. Rose Deboucha for the FISH dataset management.

References

[1] Stringer, C., Wang, T., Michaelos, M., and Pachitariu, M., "Cellpose: a generalist algorithm for cellular segmentation," *Nature Methods* **18**, 100–106 (1 2021).

[2] Gao, L., Zhang, L., Liu, C., and Wu, S., "Handling imbalanced medical image data: A deep-learning-based one-class classification approach," *Artificial Intelligence in Medicine* **108** (8 2020).

[3] Zhang, L., Yang, H., and Jiang, Z., "Imbalanced biomedical data classification using self-adaptive multilayer elm combined with dynamic gan," *BioMedical Engineering Online* **17** (12 2018).

[4] Pes, B., "Handling class imbalance in high-dimensional biomedical datasets," in [*2019 IEEE 28th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*], 150–155, IEEE (6 2019).

[5] Hardy, R., Klepich, J., Mitchell, R., Hall, S., Villareal, J., and Ilin, C., "Improving nonalcoholic fatty liver disease classification performance with latent diffusion models," *Scientific Reports* **13** (12 2023).

[6] Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., and Greenspan, H., "Synthetic data augmentation using gan for improved liver lesion classification," in [*2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*], 289–293, IEEE (4 2018).

[7] Panigrahi, I. and Zhu, R., "Comparing importance sampling based methods for mitigating the effect of class imbalance," (2 2024).

[8] Volk, O. and Singer, G., "An adaptive cost-sensitive learning approach in neural networks to minimize local training–test class distributions mismatch," *Intelligent Systems with Applications* **21**, 200316 (3 2024).

[9] Mienye, I. D. and Sun, Y., "Performance analysis of cost-sensitive learning methods with application to imbalanced medical data," *Informatics in Medicine Unlocked* **25**, 100690 (2021).

[10] Khan, A. K. A. and Malim, N. H. A. H., "Comparative studies on resampling techniques in machine learning and deep learning models for drug-target interaction prediction," *Molecules* **28**, 1663 (2 2023).

[11] Saripuddin, M., Suliman, A., and Sameon, S. S., "Impact of resampling and deep learning to detect anomaly in imbalance time-series data," in [*2022 14th International Conference on Computer Research and Development (ICCRD)*], 37–41, IEEE (1 2022).

[12] Qin, Z., Yu, F., Liu, C., and Chen, X., "How convolutional neural networks see the world — a survey of convolutional neural network visualization methods," *Mathematical Foundations of Computing* **1**, 149–180 (2018).

[13] Ho, J., Jain, A., and Abbeel, P., "Denoising diffusion probabilistic models," (6 2020).

[14] Nichol, A. and Dhariwal, P., "Improved denoising diffusion probabilistic models," (2021).

[15] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y., "Generative adversarial nets."

[16] Ho, J. and Salimans, T., "Classifier-free diffusion guidance," (7 2022).

[17] Dhariwal, P., Openai, and Nichol, A., "Diffusion models beat gans on image synthesis."

[18] Kazerouni, A., Aghdam, E. K., Heidari, M., Azad, R., Fayyaz, M., Hacihaliloglu, I., and Merhof, D., "Diffusion models in medical imaging: A comprehensive survey," *Medical Image Analysis* **88**, 102846 (8 2023).

[19] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A., "Image-to-image translation with conditional adversarial networks," in [*2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], 5967–5976, IEEE (7 2017).

[20] Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A., "Unpaired image-to-image translation using cycle-consistent adversarial networks," in [*2017 IEEE International Conference on Computer Vision (ICCV)*], 2242–2251, IEEE (10 2017).

[21] Saharia, C., Chan, W., Chang, H., Lee, C. A., Ho, J., Salimans, T., Fleet, D. J., and Norouzi, M., "Palette: Image-to-image diffusion models," (11 2021).

[22] Li, B., Xue, K., Liu, B., and Lai, Y.-K., "Bbdm: Image-to-image translation with brownian bridge diffusion models."

[23] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B., "High-resolution image synthesis with latent diffusion models," in [*2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*], 10674–10685, IEEE (6 2022).

[24] Carlini, N., Hayes, J., Nasr, M., Jagielski, M., Sehwag, V., Tramèr, F., Balle, B., Ippolito, D., and Wallace, E., "Extracting training data from diffusion models," (1 2023).

[25] Burg, G. J. J. V. D. and Williams, C. K. I., "On memorization in probabilistic deep generative models."

[26] Hur, J., Choi, J., Han, G., Lee, D.-J., and Kim, J., "Expanding expressiveness of diffusion models with limited data via self-distillation based fine-tuning."

[27] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," in [*2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*], 770–778, IEEE (6 2016).

[28] McInnes, L., Healy, J., Saul, N., and Großberger, L., "Umap: Uniform manifold approximation and projection," *Journal of Open Source Software* **3**, 861 (9 2018).

[29] Lee, W., Han, K., Harris, C. P., Shim, A. S., Kim, S., and Meisner, L. F., "Use of fish to detect chromosomal translocations and deletions analysis of chromosome rearrangement in synovial sarcoma cells from paraffin-embedded specimens," (1993).

[30] Tsukamoto, T., Kinoshita, M., Yamada, K., Ito, H., Yamaguchi, T., Chinen, Y., Mizutani, S., Fujino, T., Kobayashi, T., Shimura, Y., Inazawa, J., and Kuroda, J., "Imaging flow cytometry-based multiplex fish for three igh translocations in multiple myeloma," *Journal of Human Genetics* **68**, 507–514 (7 2023).

[31] Song, J., Meng, C., and Ermon, S., "Denoising diffusion implicit models," (10 2020).

[32] Pachitariu, M. and Stringer, C., "Cellpose 2.0: how to train your own model," *Nature Methods* **19**, 1634–1641 (12 2022).

[33] Ronneberger, O., Fischer, P., and Brox, T., [*U-Net: Convolutional Networks for Biomedical Image Segmentation*], 234–241 (2015).

APPENDIX A. BBDM

In this section, we describe the experiments, model training runs, and parameters used for the Brownian Bridge Diffusion Model (BBDM).

A.1 Experiment Setup

We evaluated the BBDM on our dataset of FISH images containing chromosomal aberrations. The goal was to generate synthetic chromosomal aberrations to augment the dataset and address class imbalance.

A.2 Training Details

The BBDM was trained using the following parameters:

- Number of diffusion steps: 1000
- Learning rate: 1.0×10^{-4}
- Batch size: 8
- Optimizer: Adam
- EMA start step: 30000
- EMA decay: 0.995
- EMA update interval: 16
- Max learning rate: 1.0×10^{-4}
- Min learning rate: 5.0×10^{-7}
- Learning rate factor: 0.5
- Learning rate patience: 3000
- Learning rate cool down: 3000
- Learning rate threshold: 1.0×10^{-4}

During training, we monitored the training loss and adjusted the hyperparameters to ensure the model converged to a state where it could generate high-fidelity synthetic samples.

A.3 Results

Figure 7 shows examples of good BBDM samples which pass the UMAP filter while Figure 8 shows bad examples eliminated by the UMAP filter.

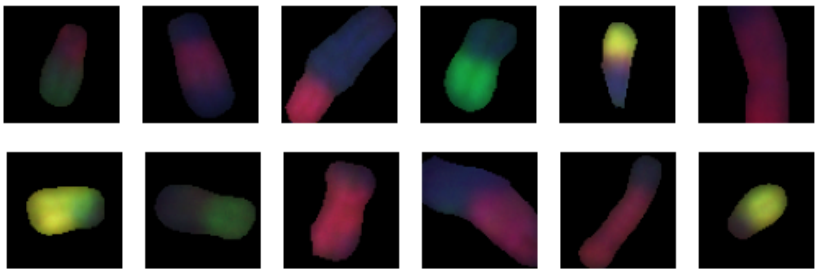

Figure 7. Example of good BBDM Synthetic Chromosomes.

APPENDIX B. RESNET

This section details the experiments, model training runs, and parameters used for the ResNet model.

Figure 8. Example of bad BBDM Synthetic Chromosomes.

B.1 Experiment Setup

We evaluated the Residual Network (ResNet) model on our dataset of FISH images for the task of classifying chromosomal aberrations. All the performance metrics (precision, recall, f1) are given as mean $\pm$ std of 10 distinct random initialization of the training runs.

B.2 Training Details

The ResNet model was trained using the following parameters:

- Number of epochs: 50
- Learning rate: 0.001
- Batch size: 32
- Optimizer: Adam

We used data augmentation techniques, including random cropping, horizontal flipping, and normalization, to enhance the training dataset and prevent overfitting.

B.3 Architecture

Below is the architecture output of the classification model. The feature-space was studied before the fully connected layer.

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (1): BasicBlock(
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (layer2): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(64, 128, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (downsample): Sequential(
        (0): Conv2d(64, 128, kernel_size=(1, 1), stride=(2, 2), bias=False)
        (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (1): BasicBlock(
      (conv1): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (layer3): Sequential(
    (0): BasicBlock(
      (conv1): Conv2d(128, 256, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
      (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```
    (downsample): Sequential(
      (0): Conv2d(128, 256, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): BasicBlock(
    (conv1): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(layer4): Sequential(
  (0): BasicBlock(
    (conv1): Conv2d(256, 512, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (downsample): Sequential(
      (0): Conv2d(256, 512, kernel_size=(1, 1), stride=(2, 2), bias=False)
      (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
  )
  (1): BasicBlock(
    (conv1): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
    (conv2): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
)
(avgpool): AdaptiveAvgPool2d(output_size=(1, 1))
(fc): Linear(in_features=512, out_features=2, bias=True)
```

APPENDIX C. CYCLEGAN

This section details the experiments, model training runs, and parameters used for the CycleGAN model.

C.1 Experiment Setup

The CycleGAN model was tested on our dataset of FISH images to perform unpaired image-to-image translation.

C.2 Training Details

The CycleGAN model was trained with the following parameters:

- Number of epochs: 200
- Initial learning rate: 2.0×10^{-4}
- Batch size: 8
- Optimizer: Adam
- Beta1: 0.5
- Cycle consistency loss weight: 10

APPENDIX D. UMAP FILTERING

This section details the use of Uniform Manifold Approximation and Projection (UMAP) for filtering synthetic samples generated by the BBDM.

D.1 Experiment Setup

We applied UMAP filtering on synthetic samples generated from our dataset of FISH images.

D.2 UMAP Configuration

The UMAP algorithm was configured with the following

parameters:

- Number of neighbors: 15
- Minimum distance: 0.1
- Number of components: 2
- Metric: euclidean