Final Year Project Report

Generating Synthetic Video Data using Variational Autoencoder

Mihaly Arpad Dani

24th March, 2025

Supervisor: Felix Riegler

SRN: 21055390

Abstract

This project addresses the challenge of limited fall-related video data for training machine learning models in real-time fall detection systems. Falls are rare, difficult to record safely, and under the data privacy laws like GDPR, restricted the use and share.

To overcome this, the project explores the use of Variational Autoencoders (VAEs) to generate synthetic fall video sequences. A baseline image-based VAE was modified to support video input using 3D convolutions and was further enhanced with skip connections to improve reconstruction quality.

Three experiments were conducted to evaluate the impact of dataset size and architecture on the quality of the generated data. Results showed that skip connections improve visual reconstruction but may lead to posterior collapse, where the latent space fails to learn meaningful representations. Contrarily, training without skip connections on a larger dataset improved latent space quality and diversity in generated samples.

While synthetic data fine-tuned on a fall detection model showed improved classification metrics, the evaluation highlighted potential overfitting. Future work includes exploring latent-to-decoder skip connections, Vector Quantised VAEs (VQ-VAEs), and training on motion-based features such as optical flow for more robust and privacy-friendly fall detection.

Acknowledgements

I would like to thank my supervisor, Felix Riegler, for his continuous support, helpful guidance during our meetings, and quick responses to all of my questions. I am also very grateful to the KTP research team, especially Dr. Na Helian, who first introduced me to Variational Autoencoders and inspired this project with her deep knowledge and enthusiasm for research. Finally, I wish to thank Kitti Pintér for proofreading my report and providing valuable feedback on clarity and structure.

Table Of Content

Glossary

Generative Adversarial Network - Generative model composed of two network (generator and discriminator) competine to generate realistic data.

Variational Autoencoder (VAE) - Generative model that learns a probabilistic latent space and capable of generating new samples, similar to the training data

Encoder - The first part of an autoencoder that compresses input data into latent representation

Decoder - The second part of an autoencoder that reconstructs the original input from the compressed latent representation

Skip Connections - A neural network design that allows outputs from earlier layers to be passed directly to later layers, helping preserve information and improve training in deep models.

Latent Space - A compressed representation space in which similar inputs are mapped to similar points. It is used by generative models to learn abstract features of the data.

Optical Flow - A method for estimating motion between two frames, by calculating the apparent movement of pixels.

Reconstruction Loss - A component of the VAE loss function that measures how accurately the decoder recreates the original input from the latent space.

3D Convolution - A type of convolution used in video or volumetric data, applying filters across time and space to extract spatiotemporal features

1. Introduction

Machine learning models need a lot of data to perform well. In many cases, collecting a lot of real-world data is hard, expensive, or impossible. One example of this appears in fall detection systems, where the goal is to identify when a person has fallen using video data. Falls are rare events, and recording them naturally is difficult. However, pretending to fall on purpose can be dangerous, so recording enough good examples to train accurate models is hard.

Another major challenge is privacy. Video recordings of people falling are considered personal data, and collecting or storing such footage raises concerns under the General Data Protection Regulation (GDPR). For this reason, many fall-related datasets are restricted or cannot be shared publicly. According to Stephenson Harwood's 2024 analysis of AI and big data law, AI systems processing personal data must follow strict rules under the GDPR, including data minimisation, a lawful basis for processing, and protection of individuals rights.

This issue became especially clear to me while working on a fall detection project as part of a Knowledge Transfer Partnership (KTP) between my university and a home healthcare company. The project aims to develop real-time fall detection. Early in the project, it became clear that the biggest obstacle was the lack of training data, particularly real fall samples. As a result, my primary responsibility within the project became generating high-quality synthetic data that mimics real falls, with the goal of improving the model's accuracy. The KTP Classifier model (Toh et al., 2024) is trained using the UP-Fall Dataset, which was recorded in a controlled environment with various fall and non-fall activities performed by human subjects (Martinez-Villaseñor et al., 2019). For the purpose of the KTP project, the dataset is simplified into two main categories: fall and non-fall.

This project explores a solution to these problems using Variational Autoencoders (VAEs). VAEs are machine learning models that can generate new, artificial data based on the patterns they learn from real examples. The goal of this project is to train a VAE that can produce realistic synthetic fall-related data, which can then be used to support the training of fall detection models. This avoids the need for sensitive personal data and can help overcome the lack of training samples.

The main objectives of the project are to study how VAEs work, prepare real fall detection data as input, train the VAE, and evaluate the quality of the generated data by fine-tuning an existing fall detection model.

Project Aim:

The Project Aim is to generate synthetic fall videos using Variational Autoencoder (VAE) model, to help balance the training data used in fall detection systems when real falls are rare.

Project Objectives:

- To study and understand the theory behind VAEs and their application to video Data.
- To adjust existing VAE architecture to process short video sequences (9 frames)
- To train the VAE model using real fall and non-fall video clips from UP-Fall Dataset.
- To Experiment with skip connection and evaluate their impact on reconstruction quality and the latent space.

Project Goals:

- Build a working VideoVAE
  - Convert 2D image-based VAE to support 3D Video Data
  - Add Skip Connections to Encoder-Decoder Architecture
- Train and evaluate Different model versions
  - Compare models with and without skip connections
- Analyse the quality of reconstructed and generated data.
- Explore how the Synthetic Data might support a fall detection classifier

This report is structured as follows:

- **Chapter 2 (Literature Review)** - An overview of existing research on Fall Detection, Synthetic data and how they are used in healthcare
- **Chapter 3 (Technical Background)** - Focuses on the concepts of Generative models like VAEs and GANs. Explains the foundational knowledge needed to understand the methods used in this project.
- **Chapter 4 (Methodology)** - Describes how the dataset was prepared, how the VAE model was adapted and trained, and the training process of different experiments.
- **Chapter 5 (Results and Evaluation)** - Presents and compares the outcomes of each experiment.
- **Chapter 6 (Conclusion)** - Summarises what was achieved, the project's limitations and potential improvements and ideas for future work.
- **Chapter 7 (Project Management Review)** - Reflect on my strengths and Weaknesses as a Project manager and my time management

2. Literature Review

This literature review studies the key concepts and research for fall detection systems and the use of synthetic data in healthcare. It begins by discussing the Knowledge Transfer Partnership (KTP) project that motivates this study, highlighting the challenges of getting real-world fall data due to privacy, ethical, and practical limitations. Review how synthetic data is used to handle these challenges in healthcare, allowing researchers to simulate data that reflects real data. Then, the focus goes to Variational Autoencoders (VAEs) as a technique for generating realistic synthetic video data and how architectural enhancements like skip connections may improve the quality of generated samples.

2.1 The KTP Project and Fall Detection

The fall detection project is part of a Knowledge Transfer Partnership (KTP) between my university and a home healthcare company. The aim is to build a real-time fall detection model that can identify when someone has fallen using video data (Toh et al., 2024). To train such a model, a large amount of high-quality data is needed.

The dataset used in the project (Martinez-Villaseñor et al., 2019) consists of video recordings where subjects perform various fall and non-fall activities. These videos were recorded at 18 frames per second (fps), and each frame was saved as a PNG image.

To make the data more suitable for training machine learning models, the preprocessing consists of two main components:

- Each video is broken down into smaller segments using a sliding window approach. Each window contains 18 frames (one second of footage), and there is a 50% overlap between consecutive windows, creating multiple overlapping windows for the continuous monitoring of sequential movements.
- Optical flow is applied to extract motion features from the video frames. The research uses the Farneback method, which estimates the motion between consecutive frames by approximating the neighbourhood of each pixel with quadratic polynomials and comparing these polynomials to derive displacement fields. This technique captures the direction and magnitude of movement between frames, highlighting the spatiotemporal dynamics of human motion that are critical for fall detection.

2.2 Synthetic Data in Healthcare

In healthcare and many other fields, access to real data is often restricted due to privacy concerns, legal regulations, or simply because collecting it is difficult and expensive. This is especially true for sensitive areas like videos of people falling, which are considered personal data under the regulations of the General Data Protection Regulation (GDPR).

Many researchers use synthetic data to handle this. In healthcare, synthetic data can be used to create realistic patient records, simulate disease spread, or build and test health-related software tools (Gonzales et al., 2023). Synthetic data is already having a positive impact by making valuable data available without risking privacy.

Here are a few examples of how synthetic data is used in healthcare today:

- To research how a new health policy might impact hospital visits.
- In education, where students can learn using realistic but anonymised data.
- For data linking, where synthetic data is used to test if different datasets can be correctly matched.

2.3 Benefits and Challenges of Synthetic Data

The biggest advantage of synthetic data is that it makes it possible to train models when real data is hard to get. For example, in fall detection, there are plenty of videos showing people walking or sitting but very few real falls. It's unsafe to ask people to fall on purpose, and natural falls are rare, so getting enough data is very hard.

Synthetic data solves this problem by creating new examples that look realistic and follow similar motion patterns without putting anyone at risk or violating privacy laws. This makes it a powerful tool in areas like healthcare and safety (Lu et al., 2024).

However, there are some challenges. If the synthetic data doesn't accurately reflect real-world variation, the model might not learn the right patterns. It may perform well on test data but fail in real-life situations. Generating good synthetic data requires well-designed models, a poorly trained model might generate data that damages training instead of helping.

2.4 Synthetic data generation using VAEs

VAEs are widely used in synthetic data generation because of their ability to create new, realistic data samples. In fields like healthcare, where privacy and data scarcity are common challenges, VAEs have been used to generate synthetic patient records, medical images, and time-series data. For instance, in healthcare applications, VAEs can generate anonymised versions of patient data that preserve statistical patterns without revealing personal information. This is especially useful for training models when access to real patient data is limited. (Nikolentzos et al., 2023)

In fall detection, real-world fall data is difficult to collect due to safety and ethical concerns. VAEs can be used to generate synthetic fall data based on the limited real samples available. Research such as that by Chen, Wang, and Yang (2021) focuses on using pose-based data for fall detection, showing how models benefit from more diverse and motion-rich datasets.

2.5 Relevance to This Project

In this project, I focused on generating synthetic fall data using Variational Autoencoders (VAEs) to support fall detection model training. Since real fall data is rare and sensitive, synthetic data helps increase the diversity and quantity of the training examples. This is especially useful when the training dataset is imbalanced, like in the UP-Fall Dataset.

The VAE model introduced by Kingma et al. (2013) is designed for data points like vectors or images, so I adapted it to handle video sequences by modifying the architecture to process frame sequences and learn temporal information (how things move over time).

To improve the model further, I experimented with adding skip connections into the model architecture. Skip connections or Residual Blocks became more common (Xu et al., 2024) in deep learning through the ResNet architecture by He et al. (2015). These connections allow the model to reuse important features from earlier layers, and this helps preserve motion details across frames.

The synthetic data generated by this modified VAE is then combined with the original dataset to train the fall detection model. The aim is to improve model performance by adding more examples, especially of falls, which are otherwise underrepresented in the dataset.

3. Technical Background

Presenting the foundational knowledge necessary to understand the methods used in this project. Introducing generative models, machine learning models capable of creating new, realistic data samples. Two types of generative models are focused on: Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). The focus is then narrowed to VAEs, as they form the core of this project. Key concepts like encoder-decoder structure, latent space, reparameterisation trick, KL divergence, and the VAE loss function are explained in detail. Also, the chapter covers the importance of optical flow for capturing motion in video sequences and introduces skip connections, a deep learning technique used to improve reconstruction quality.

3.1 Generative Models

Generative models are a class of ML models that learn the underlying probability distribution of a dataset and can generate new data samples that look like the original data. In

other words, after training on real examples, a generative model can produce synthetic data similar to those examples. Most known generative models include Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs), as well as autoregressive and normalising flows (Li et al., 2024). The key benefit of generative models is that they are capable of creating new (previously unseen) data. This is very useful for Synthetic Data Generation. For example, if collecting real video data is difficult or some events (like human falls) are rare, a generative model can be used to produce additional realistic video samples after it is trained on a varied dataset (Lu et al., 2024).

3.2 Generative Adversarial Networks

GANs are one of the most popular types of generative models introduced by Goodfellow et al. (2014). A GAN has two types of neural networks. A Generator and a Discriminator that compete with each other. The Generator tries to create fake data (image or video clip) that looks real, while the Discriminator examines data and tries to tell real vs generated inputs. During training, the generator is optimised to fool the discriminator, and the discriminator is optimised to become better at spotting fake data. Through this adversarial process, the generator gradually learns to produce highly realistic outputs (Goodfellow *et al.*, 2014). GANs demonstrated impressive results in recent years in generating images and have extensions for video generations as well. Their biggest challenge is that during the training, careful balance is required between the two networks. Issues like mode collapse (producing limited varieties of outputs) and training instability are common issues. Different techniques are used to stabilise the GAN training, like special loss functions and network architecture.

3.3 Variational Autoencoder

VAEs, on the other hand, use a different approach introduced by *Kingma and Welling in 2013,* and this project is mainly built on their work. VAE consists of an Encoder and a Decoder. The Encoder compresses input data (Image or Video) into a latent representation in a set of parameters that define a probability distribution (usually Gaussian) in the latent space. Instead of directly encoding the input into a fixed vector, the encoder outputs the mean and variance of the distribution. A latent vector z is then sampled from this distribution using a technique called reparameterisation trick, which allows backpropagation through the sampling step. The latent representation is a downsampled, lower-dimensional version of the original data that captures the most important and informative features. The Decoder then reconstructs the input data from this compressed form, allowing the model to generate new data by sampling from the latent space, enabling the VAE to create outputs that follow the data distribution without being exactly copied from the training dataset.

3.3.1 How Vae Works

The encoder network processes the input data x (an image) and learns to represent it using two vectors: one for the mean $\mu(x)$, and another for the variance $\sigma^2(x)$. These define a Gaussian (bell-shaped) probability distribution in the latent space. Instead of compressing the input into a single fixed point, the encoder outputs parameters that describe a whole distribution; this is a key difference between VAEs and traditional autoencoders. This allows the model to treat the latent representation as a distribution rather than a fixed point because it outputs both mean and variance for each latent dimension. This probabilistic approach enables the model to understand that similar inputs might have slightly different outputs. By learning a range (mean and variance) instead of just one value, the model can create new examples that are slightly different but still believable. This makes it better at handling variation and producing new, realistic samples from the data it has learned.

From this distribution, a latent vector z is sampled, which is a compressed representation of the input. The decoder will later use this vector z to try and reconstruct the original input. However, sampling directly from a distribution during training is challenging because it breaks the flow of gradients needed for learning.

To solve this, Variational Autoencoders use a technique called the reparameterisation trick. Instead of sampling z directly from the distribution, a small amount of random noise is introduced by sampling a vector $\epsilon \sim \mathcal{N}(0, I)$, where each element of ϵ is drawn from a standard normal distribution with mean 0 and variance 1.

To compute the latent vector :

$$z = \mu(x) + \sigma(x) \cdot \epsilon$$

This equation transforms the random noise ϵ using the learned mean and standard deviation, resulting in a sampled latent vector z that still reflects the input but includes controlled randomness.

This makes the sampling operation differentiable, allowing gradients to flow through μ and σ during backpropagation. As a result, the model can be trained using standard optimisation techniques like gradient descent.

During Training (**Reconstruction**):

The z Calculated with the input data mean $\mu(x)$ and variance $\sigma(x)$ multiplied by the random ϵ

$$z = \mu(x) + \sigma(x) \cdot \epsilon$$

During generation (**sampling new data**):
Since there is no input data anymore, and the encoder is skipped, the z is sampled directly from latent distribution:

$$z = \epsilon$$

Where $\epsilon \sim \mathcal{N}(0, I)$

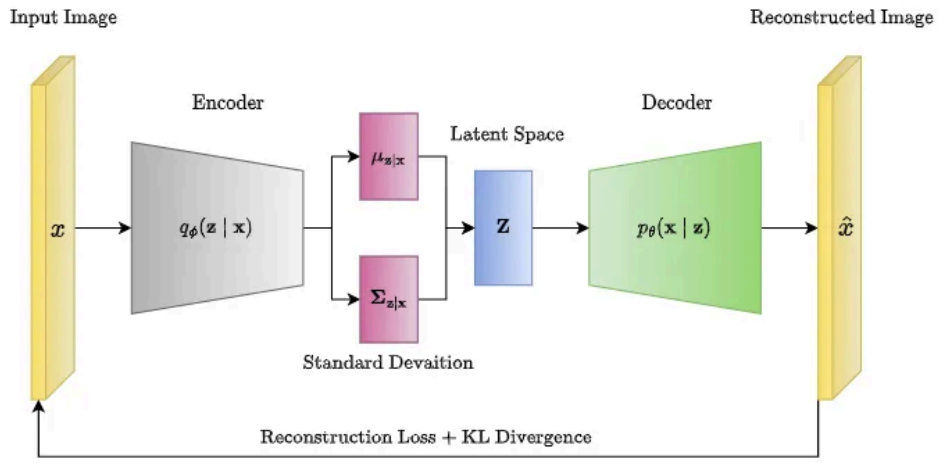


Figure 1: Network Architecture for a Variational Autoencoder

3.3.2 Latent Space, Sampling, KL divergence

After the encoder processes an input image x, it outputs two vectors: the mean $\mu(x)$ and the variance $\sigma^2(x)$, which defines a Guassian distribution in a lower-dimensional space, called Latent Space. This latent space is a compressed representation of the dataset and contains the key patterns or features from the inputs.

The Purpose of the latent space is to allow the mode to Encode Information about the input data in a compact and meaningful way. Similar inputs, for example, two different images of a person falling, will be mapped to similar regions of the latent space, while different inputs like sitting, will be mapped farther apart. This structure allows the VAE to learn relationships and variations between data samples.

Once the model has learned this space, it is possible to use interpolation between two points in the latent space. For example, moving smoothly from one latent vector to another and decoding the intermediate vectors can generate smooth transitions between images. Resulting in turning one pose (Standing) into another (Lying).

During Training, unfixed points are used for each input, but a sampled latent vector z from the Guassian distribution by $\mu(x)$ and $\sigma(x)$, using the reparameterisation trick:

$$z = \mu(x) + \sigma(x) \cdot \epsilon \text{ where } \epsilon \sim \mathcal{N}(0, I)$$

This sampling introduces a small amount of randomness into each training example, which makes sure that the model learns general rules of flexible representation instead of memorising exact details.

Later, when generating new synthetic data, there is no Input image to pass through the encoder, so directly sampling from the latent space using Standard Normal Distribution

$$z \sim \mathcal{N}(0, I)$$

Then this z is passed to the decoder, which generates the new output sample. This works only if the latent space learned during the training is close to the standard normal distribution.

To ensure that the encoder outputs distributions that are close to this standard normal prior ($\mathcal{N}(0, I)$), a special penalty term is included into the loss function called Kullback-Leibler Divergence.

This KL divergence is written as:

$$D_{KL}(q_\phi(z|x) \parallel p(z)) = \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\right)]$$

Here, $q_\phi(z|x)$ is the encoder's learned distribution for input x (defined by μ and σ^2), and $p(z)$ is the prior distribution, usually set to $\mathcal{N}(0, I)$. The formula above measures how different these two distributions are. If the KL divergence is small, it means the encoder is producing distributions that look very similar to the standard normal. This is a good sign, as it keeps the latent space smooth and compatible with the sample. So during training, the aim is to minimise the KL divergence.

3.3.3 VAE Loss Function (Reconstruction + KL)

The VAEs are Trained by minimising a special loss function that balances two main goals: encourages $\hat{x}$ (reconstructed input) to be as similar as possible to the original x, and to ensure that the latent space is smooth and follows standard normal distribution. To Achieve this the Loss Function have two main components:

- **Reconstruction Loss** which measures how close the reconstructed $\hat{x}$ (Generated by the decoder) is to the original input x. If the Reconstruction is poor, the model will be penalised. This typically measured by Mean Squared Error (MSE) or Binary Cross-Entropy (BCE), depending on the data type:

  For continuous data (pixels values between 0 and 1), often the BCE is used:

$$\mathcal{L}_{\text{recon}} = -\sum [x \cdot \log(\hat{x}) + (1 - x) \cdot \log(1 - \hat{x})]$$

  For grayscale or colour images, MSE is often used:

$$\mathcal{L}_{\text{recon}} = \|x - \hat{x}\|^2$$

- **KL Divergence Loss** ensures that the latent variables produced by the encoder follows a standard normal distribution $\mathcal{N}(0, I)$ which measures how different the learned distribution (defined by ($\mu(x)$ and $\sigma(x)$) is from the standard normal prior.

$$D_{KL}(q_\phi(z|x) \parallel p(z)) = \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\right)]$$

Each element μ_i and σ_i refers to a signal dimension of the latent vector. This Penalises: High values of μ_i (Shifts away from zero), Large or very small σ_i (spreading too far or collapsing), or the Distributions that are too different from the normal distribution. By minimising this term, the latent space stays close to standard Guassian, so the sampling is smooth.

By adding the two Loss together the results are called Evidence Lower Bound (ELBO). The Reconstruction Loss Part can Vary depending on the chosen Reconstruction formula and the output activation (Sigmoid, Tanh)

$$\mathcal{L}_{\text{VAE}} = \underbrace{\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]}_{\text{Reconstruction Loss}} - \underbrace{D_{KL}(q_\phi(z|x) \| p(z))}_{\text{KL Divergence}}$$

3.5 Optical Flow and Motion encoding

Optical Flow is a technique used in Computer Vision to estimate the motion of the object between consecutive video frames. It calculates how pixels move from one frame to the next, giving a motion map that shows the direction and speed of the movement at each point of the

image. In fall detection, this is especially useful because falls involve movements that can be captured by analysing the flow motion.

Instead of Using raw RGB frames, which include a lot of background and irrelevant information, the KTP project classifier model uses optical flow as input. This helps the model to focus on the changes over time, rather than the static details like clothing, room or environment. This results in motion based features which are more reliable for identifying falls.

3.6 Skip Connections in Deep Learning

Skip connections are links in neural networks that skip over one or more layers and give later layers direct access to earlier layer outputs. In the ResNet Architecture, it is shown how skip connections can enable the training of deep networks effectively (He et al. 2015).
Skip connections have two main benefits:

- Better gradient flow: By providing an alternate path for gradients during back-propagation, skip connections help reduce the vanishing gradient problem. For example, in ResNet, the output of layer n is added to the output of a much later layer m. This means the loss gradient can flow directly back to layer n without being multiplied by all the weights between n and m. This allows networks with many layers to learn because even the early layers get a strong supervision signal. In practice, networks with skip connections (residual networks) train faster and can achieve higher accuracy as they get deeper.
- Feature reuse: Skip connections let the network combine low-level features with high-level features. An early layer might capture fine details (edges, textures), and a later layer might capture abstract concepts (object shapes or motions). By skipping, the model can use both kinds of information. This is especially useful in generative models or segmentation tasks. For example, ResNet uses skip connections from encoder layers to decoder layers so that the decoder can utilise high-resolution details from the encoder. In VAEs, skip connections could similarly help preserve details or temporal consistency by channeling information from early layers directly into the reconstruction process.

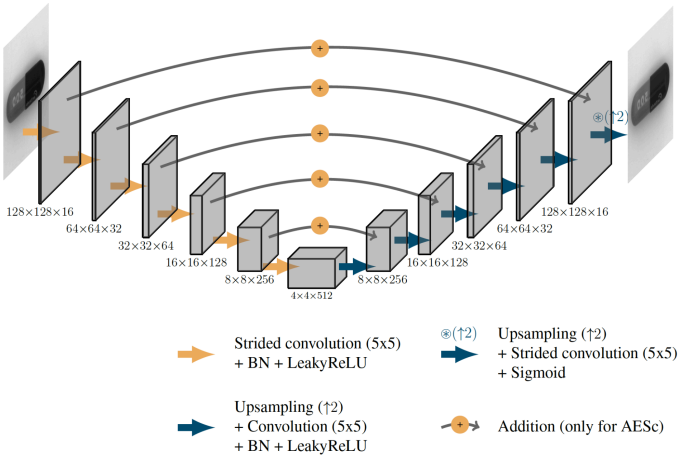


Figure 2: Illustration of an Autoencoder with skip connections between the encoder and decoder

4. Methodology

This chapter outlines the practical steps to build and evaluate a Variational Autoencoder (VAE) for generating synthetic video data related to human falls. First, how to prepare the UP-Fall Dataset by putting into short video windows and resizing the data for VAE training. The original VAE architecture was modified to handle 3D video input by replacing 2D convolutions with 3D convolutions.

Skip connections were introduced to improve detail retention during reconstruction. The training process used a loss function combining reconstruction and KL divergence, with early stopping and visual outputs to track performance. Three experiments were conducted to evaluate the impact of dataset size and skip connections on the quality of reconstructed and generated video samples

4.1 Data Preparation

The Goal for this stage was to prepare the UP-Fall Dataset to make it suitable for my Variational Autoencoder. The UP-Fall dataset contains individual images (frames) taken from videos. These frames represent human activities such as walking, sitting, falling, lying, etc. All activities are captured from two different camera angles and multiple sensors. The dataset is organised in groups on short segments called “windows”, each 0.5 seconds long, containing exactly 9 frames. Importantly, each window represents a consistent activity, meaning the activity label stays the same within each window.

Each Frame was resized to a standard size of 128x128 pixels, and the windows were saved in a Numpy array shape (number_of_windows, 9, 128, 128, 3). Initially, only the window

containing fall was filtered, but since this didn't provide enough data to train the VAE efficiently, the entire activity video, including all labels,were later included to increase the training dataset size. Example, a Fall Activity starts with windows Standing (Label 7), then the fall happens (1-5 labels), then finishes with Laying (Label 11)

Activity ID	Description	Duration (s)
1	Falling forward using hands	10
2	Falling forward using knees	10
3	Falling backwards	10
4	Falling sideward	10
5	Falling sitting in empty chair	10
6	Walking	60
7	Standing	60
8	Sitting	60
9	Picking up an object	10
10	Jumping	30
11	Laying	60

Figure 3: Description of each Activity type and their duration in seconds

4.2 Model Architecture

The base model used in this project is a simple implementation of the original Variational Autoencoder (VAE) architecture, replicating the design introduced by Kingma and Welling (2013). A publicly available GitHub repository (Subramanian, 2020), was used as a starting point, and the architecture was modified to support video input through 3D convolutions and additional enhancements.

4.2.1 Adapting Input and Output Dimensions for Video Data

The first modification involved redefining the input and output from 2D images to 3D video sequences. In the Vanilla VAE, the model processes a single image with dimensions [batch_size, channels, height, width]. For the VideoVAE, this was adjusted to handle sequences of frames represented as [batch_size, channels, frames, height, width], where the additional frames dimension account for the temporal aspect of the data. Specifically, the model was designed to process windows of 9 frames, each resize to 128x128 and 3 color channels, which resulted in an input shape of [batch_size, 3, 9, 128, 128].

By adapting the model to handle 3D inputs and outputs, it can learn and reconstruct not just the spatial patterns, but also how these patterns change over time.

4.2.2 Replacing 2D Convolutions with 3D Convolutions

The 2D Convolution Layers were replaced in both Encoder and Decoder with 3D Convolutional layers. In the Vanilla VAE, 2D convolutions apply filters across the height and width of an image, capturing spatial features like edges and textures within a single frame. For the Video, 3D convolutions were used with kernel spanning [depth, height, width], where depth corresponds to the temporal dimension (frames). This makes possible to filters to slide across both spatial and temporal axes, extracting features that combine spatial details (shape of the person) with temporal changes (motion of falling)

This is a crucial change to make because 2D convolutions, when applied frame by frame, fail to model the continuity and dependencies between frames, treating each frame independently. While the 3D convolutions look at the whole video sequence at once, they capture both what each frame shows and how things change over time. This is why the model can learn the motion and dynamics of a fall

4.2.3 Modifying the Encoder Architecture

As the model uses 3D convolutions, small changes were made in the encoder to work properly with video clips. The original Vanilla VAE model, the encoder reduced the image size by half at each layer until it was small enough to flatten and send into the latent space. The same idea was used, but to make sure that the time dimension (9 frames) stayed the same during the encoding, I used a stride of (1, 2, 2) (Depth, Height, Width) which reduced the height and width by half, but not the number of frames.

At the end of the encoder, the output shape is [batch_size, 512, 9, 4, 4], which still keeps the 9 frames. This design keeps the full motion sequence while reducing the amount of spatial detail. The idea is to compress the video just enough to make it manageable, without losing the frame motions.

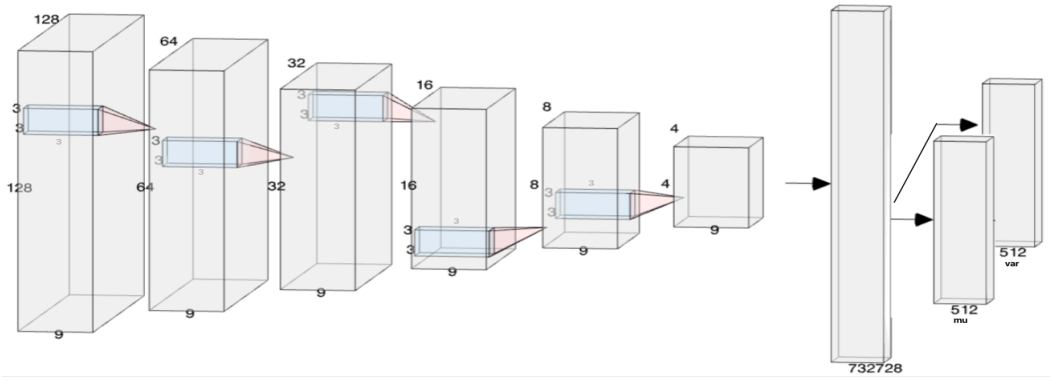


Figure 4: Illustration of my VideoVAE Encoder Architecture

4.2.4 Updating the Latent Space

After the encoder processes the video, the next step is to flatten the result so it can be turned into a compressed embedding called latent vector. The tensor [batch_size, 512, 9, 4, 4] is flattened into a long vector size 73,728. Two separate layers then turn this vector into the mu and log_var vectors, which represent the learned probability distribution. This part is mostly the same as in the Vanilla VAE.

For decoding, the process was reversed. A linear layer converts the latent vector batch into the shape [batch_size, 512, 9, 4, 4] so the decoder can reconstruct the video from there. The reason for keeping the probabilistic part (mu and log_var) is that it lets the model generate many different outputs from similar inputs (sampling), which is great for creating diverse training data.

4.2.5 Building a decoder to Recreate the Video

The decoder is the part of the model that rebuilds the video from the latent vector. Starting with [batch_size, 512, 9, 4, 4], a series of 3D transposed convolutions was used to increase the spatial dimensions back to 128x128 while keeping the number of frames at 9. The last layer outputs [batch_size, 3, 9, 128, 128], which is exactly the same shape as the input.

Sigmoid activation function was used at the end to scale the pixel values between 0 and 1. This helps the model keep its outputs stable and predictable. This stage is important because it allows the model to take compressed data and turn it back into full videos.

4.2.6 Adding Skip Connections for better Detail

To improve the quality of the reconstructed video, as an experiment, skip connections are added between the encoder and decoder. Skip connections, or residual connections, directly pass feature maps from earlier layers in the encoder to corresponding layers in the decoder. This technique is used in the U-Net and ResNet (He et al. 2015), where low-level spatial details lost during downsampling are reintroduced during upsampling.

In my model, skip connections were added from the output of each encoder block (except the last one) to the corresponding decoder block:

- The output of encoder_block1 is added to decoder_block4
- The output of encoder_block2 is added to decoder_block3
- The output of encoder_block3 is added to decoder_block2
- The output of encoder_block4 is added to decoder_block1

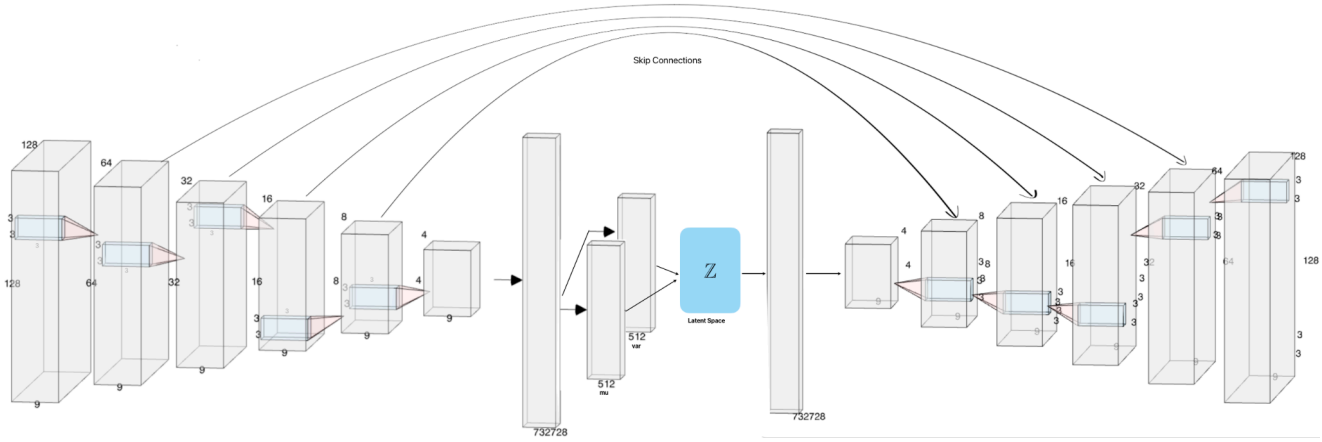


Figure 5: Illustration of my VideoVAE Encoder and Decoder Architecture with Skip Connections

This design helps to remember important spatial details (Edges, Shapes etc) which otherwise may be lost or very blurred during the encoding process. Injecting these features back into the decoding process, the model gets a little help to reconstruct the frames.

4.3 Training

The entire training phase was designed to easily switch between trained model versions and experiments. For each version, the configuration and scripts are saved, so by simply changing the version number at the beginning, a new one is created (if it doesn't already exist) or the existing configuration, model architecture, and all necessary utilities for that version are loaded.

The training process aims to optimise the performance of the VideoVAE model using different configurations and experimental setups. Implemented in PyTorch, the pipeline includes automated plot generation and result saving at each epoch, as well as automatic saving of the best-performing model.

The model was trained using a combination of reconstruction loss (Mean Squared Error) and Kullback–Leibler (KL) divergence loss, which balances accurate reconstruction of input data and regularisation of the latent space. The total loss is calculated as:

$$\text{Total Loss} = \text{Reconstruction Loss} + \beta \cdot \text{KL Divergence}$$

The β parameter is used to scale the KL loss. This beta formulation allows better control over the organisation of meaningful features in the latent space.

Training Parameters

The model training is controlled with the next configuration parameters:

latent_dim: The size of the latent vector which represents the compressed information from the flattened vector.

batch_size: Number of video samples processed together in one forward pass during the training

num_epochs: Total number of times the full training dataset is passed through the model during the training. Across all versions, it is set to **50**

learning_rate: Controls the step size at each iteration while getting closer to the minimum of the loss function

beta_start, beta_end: In the next experiments, it is set to **1** (No effect).

weight_decay: Regularisation that helps to prevent overfitting by penalising large weights in the model; it is used in the Adam normaliser.

early_stop_patience: Number of epochs to wait without improvement on validation loss before triggering early stopping. Across all versions, it is set to **5**

train_ratio: The Proportion of the dataset used for training. Across all versions, it is set to **0.7**

val_ratio: The Proportion of the dataset used for training. Across all versions, it is set to **0.2**

test_ratio: The Proportion of the dataset used for training. Across all versions, it is set to **0.1**

Optimiser: Adam was the default optimiser.

Features and Their Impact

Early Stopping

Early stopping helps prevent overfitting by monitoring the validation loss and stops training when no improvement is made for a defined number of epochs. This makes sure the model generalises better and saves training time.

Beta Scheduling

The scheduling of the β parameter allows the model to focus more on reconstruction in the early stages (if β is low), and gradually shift to learning a well-structured latent space (as β increases). This dynamic balance encourages both high-quality reconstructions and useful latent representations. However, in some experiments, this is disabled by setting beta to 1.

Evaluation Visualisations

During training, visual plots are generated, including:

- Reconstruction plots from the validation set
- Random video samples generated from the latent space
- Latent space visualisation and clustering
- t-SNE projections to see the structure of the latent space
- GIFs summarising the progress visually, when training is finished.

Experiment V10.00

Skip Connections: **Off** Falls Only: **True**

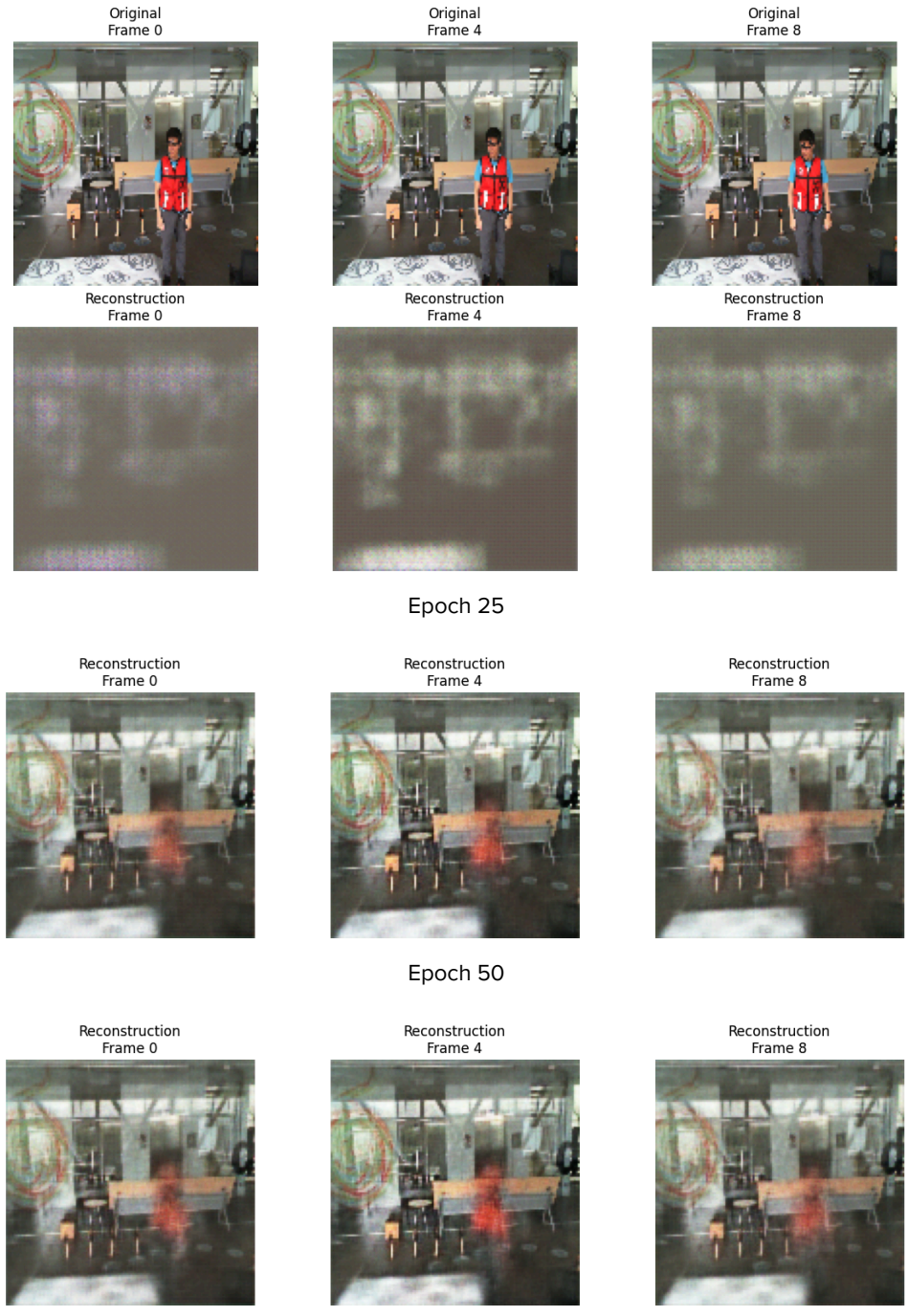


Figure 6: The first row shows original video frames (Frames 0, 4, and 8), second row shows their corresponding reconstructions generated by the VideoVAE model after first training epoch, third row reconstructed frames at epoch 25, last row at epoch 50.

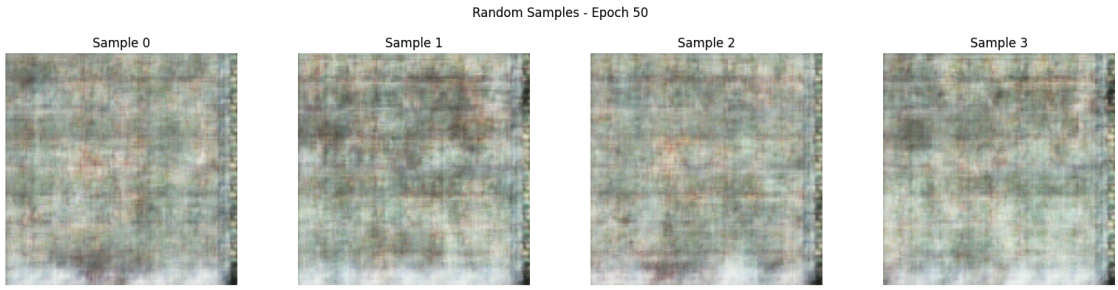


Figure 7: Random samples generated from the latent space by the VideoVAE model after 50 epochs. These samples represent video frames synthesised without conditioning on any input.

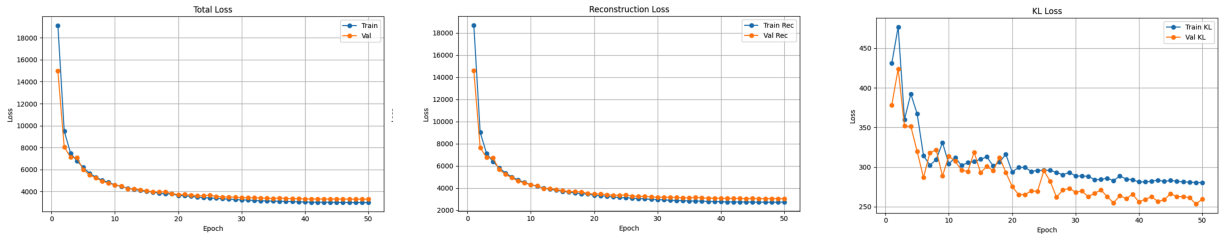


Figure 8: Training and validation curves for total loss, reconstruction loss, and KL divergence loss over 50 epochs. The model shows stable convergence, with validation loss closely tracking training loss.

In this Experiment, the VideoVAE model was trained only on video clips showing people falling (label 1-5), using all 632 available windows of 9 frames each. Skip connections were turned off, and the goal was to see how well the model can learn and recreate fall actions.

Reconstruction:

Figure 6 Shows Three original frames (0, 4, and 8) and how their reconstructions improve over time. At epoch 1 the Reconstructions are very blurry. The model hasn't learned much yet, and the person is not visible at all. At Epoch 25, the background details begin to appear, but the person is still unclear, blended into the background. At epoch 50, there's no big improvement. The background looks slightly sharper, and the person has a bit more colour, but the figure is still very blurry.

Generated Samples

Figure 7 Shows new frames created by the model from latent space. The samples are very blurry and don't show any clear background or person. Some colours and shapes give small hints, like

the white mattress at the bottom or the blurry orange patch in the middle, which is likely the subject's high-visibility vest.

Training Results

Figure 8 Shows the training and validation loss curves. The loss decreases steadily, which means the model is learning. The early stopping counter was first triggered at epoch 18, but since the early_stop_patience was set to 5, the training continued and successfully finished at epoch 50.

Final Loss:

- **Train Loss:** 3005.61 (Rec: 2725.43, KL: 280.18)
- **Val Loss:** 3299.37 (Rec: 3039.87, KL: 259.49)

Experiment V10.10

Skip Connections: **ON** Falls Only: **True**

Figure 9: The first row shows original video frames (Frames 0, 4, and 8), second row shows their corresponding reconstructions generated by the VideoVAE model after first training epoch, third row reconstructed frames at epoch 25, last row at epoch 50.

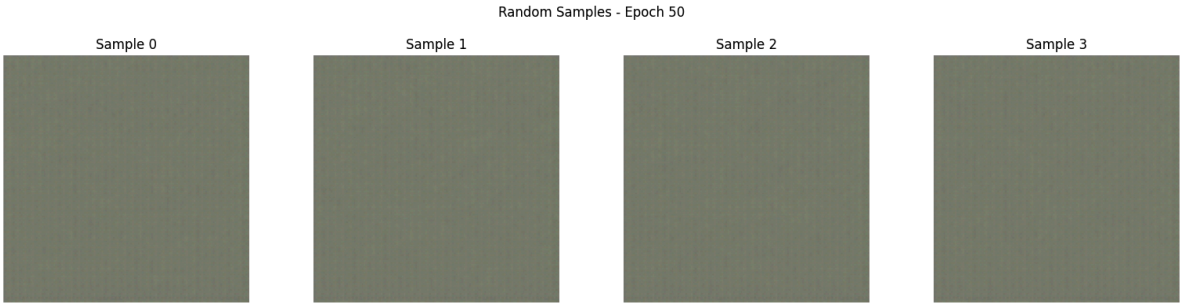


Figure 10: Random samples generated from the latent space by the VideoVAE model after 50 epochs.

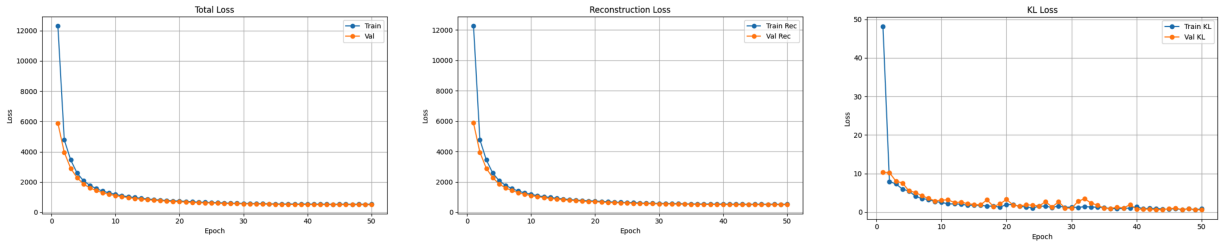


Figure 11: Training and validation curves for total loss, reconstruction loss, and KL divergence loss over 50 epochs.

In this version, the VideoVAE model was trained only on fall-related clips (labels 1–5), just like in V10.00, but this time skip connections were turned ON. These connections allow the decoder to reuse features from the encoder, helping the model retain important visual details.

Reconstruction Progress

Figure 9: Shows Three original frames (0, 4, and 8) and how their reconstructions improve over time. At epoch 1, the reconstructions are already showing clear structures, the person is visible, though still blurry. This is a big improvement compared to the previous experiment without skip connections, where the person wasn't visible at all. This shows that important details are preserved and not lost early in the network, thanks for the skip connections. At Epoch 25, both the subject and the background become much clearer. The red vest, the shape of the person, and parts of the environment are well defined. By epoch 50, the reconstruction becomes slightly sharper, but there is no significant visual improvement compared to epoch 25.

Random Samples

Figure 10, shows samples generated from latent vectors at epoch 50. These samples appear as flat, noisy patterns with no structure. Meaningless visual noise.

Training Summary

Figure 11 shows the training and validation loss curves for total, reconstruction and KL divergence loss. The first early stopping was triggered at epoch 45, but after that the model continued to improve, so it trained until epoch 50.

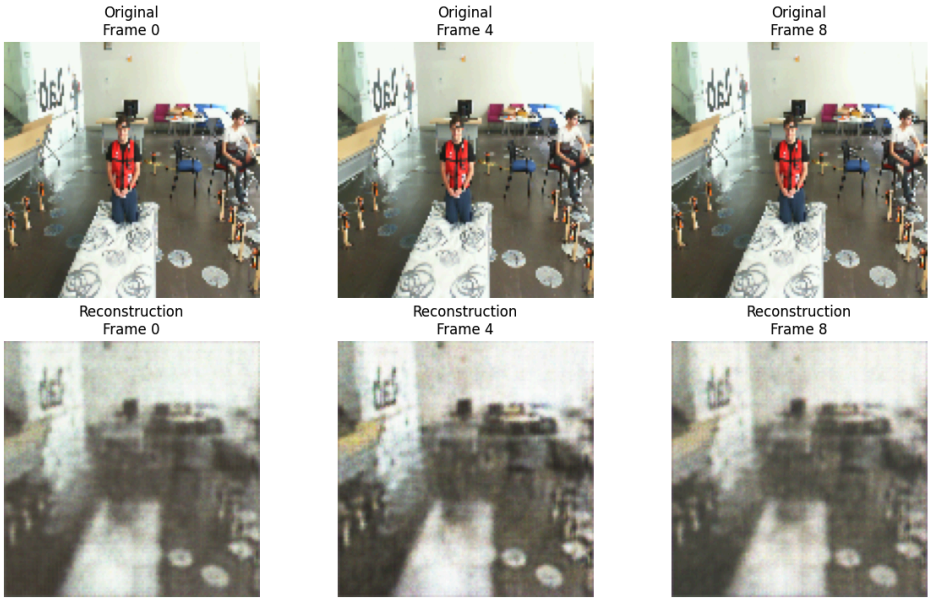
Final Loss:

- **Train Loss:** 531.47 (Rec: 530, KL: 0.74)
- **Val Loss:** 505.26 (Rec: 504.37, KL: 0.89)

Experiment V10.02

Skip Connections: **Off** Falls Only: **False**

Epoch 1



Epoch 25

Epoch 50

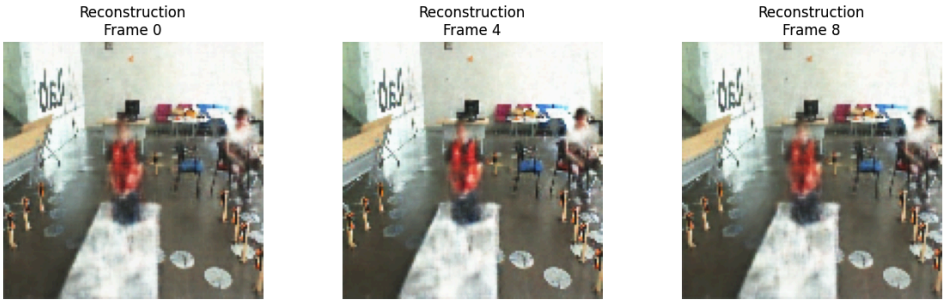


Figure 12: The first row shows original video frames (Frames 0, 4, and 8), second row shows their corresponding reconstructions generated by the VideoVAE model after first training epoch, third row reconstructed frames at epoch 25, last row at epoch 50.

Figure 13: Random samples from latent space at epoch 1, 25 and 50

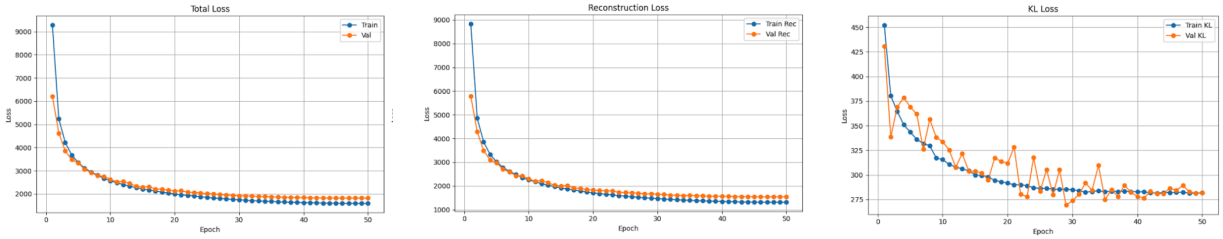


Figure 14: Training and validation curves for total loss, reconstruction loss, and KL divergence loss over 50 epochs.

In this Version, the model was trained on both Fall and Non-Fall activities using a much larger dataset (4,588 windows, each with 9 frames). Skip connections were turned Off, So the decoder had to rely only on the compressed latent feature to reconstruct the input.

Reconstruction Progress

Figure 12 Shows Three original frames (0, 4, and 8) and how their reconstructions improve over time. At Epoch 1, the background is already fairly well captured, walls, furniture, floor patterns are visible. However the person is missing, which shows that fine details are not captured early on. At epoch 25, the reconstruction improves a lot. The subject begins to appear, but very blurry. By the epoch 50, the person became sharper, but still not perfect, even though the model saw thousands of frames.

Random Samples

Figure 13 Shows a big improvement in the quality of generated samples. At epoch 1, the samples are just fuzzy noise with no structure. At epoch 25, some shapes and colours appear. By the epoch 50, some samples showed almost completed background scenes, with almost perfect layout and colours. While the person still not really visible, the background generation is much better with bigger dataset, showing that the model starts to learn some spatial context in the latent space

Training Curves

Figure 14, The KL divergence validation loss shows frequent spikes and jumps, especially in the first half of the epochs. Meaning some instability in how the model learns to match the latent distribution to the normal prior.

Final Loss:

- **Train Loss:** 1593.55 (Rec: 1311.84, KL: 281.71)
- **Val Loss:** 1824.46 (Rec: 1542.64, KL: 281.82)

5. Results and Evaluation

5.1 VAE Results and Evaluation

This section presents and critically evaluates the outcomes of the three VideoVAE experiments: V10.00 (Falls only and no skip connections),
V10.10 (Falls only, with skip connections),
V10.02 (Falls and non-falls, no skip connections).

Each experiment was trained for 50 epochs, and the results were evaluated using reconstructed outputs, randomly generated samples from the latent space, and training curves for total loss, reconstruction loss, and KL divergence loss.

5.1.1 Visual Analysis of Reconstructed Data

Experiment **V10.00** shows poor reconstruction performance throughout the training. At epoch 1, the model fails to reconstruct any meaningful features the background and person are completely blurred. By epoch 25, some elements of the background begin to appear, but the person remains nearly invisible. Even at epoch 50, there's no significant improvement: while the background becomes slightly sharper, the subject is still just a coloured blur.
In contrast,

Experiment **V10.10**, where skip connections were enabled, shows clear improvement. Even at epoch 1, the person is already visible (though blurry), suggesting that the skip connections help retain spatial features from early layers, but the frames are not colored. By epoch 25, both the subject and the background are much clearer. However, from epoch 25 to 50, there is no major visual improvement, showing the model may reach its learning capacity.

Experiment **V10.02**, trained on a much larger dataset containing both fall and non-fall clips (4,588 windows), performed better in background reconstruction despite not using skip connections. At epoch 1, the walls, floor, and general room layout are already visible, but the person is missing. Over time, the model does learn to reconstruct the subject, but even by epoch 50, the person is still noticeably blurry. This suggests that while the model generalises well to background and static elements, it struggles to reconstruct subjects in motion. Without skip connections, high-frequency details, like movements, may be lost during the encoding process, limiting the decoder's ability to recover dynamic information.

5.1.2 Evaluation of Latent Space Generation

In Experiment **V10.00**, the random samples at epoch 50 are essentially visual noise. There is no meaningful structure or shape; only hints of colours orange for high-vis vests in the middle and faint patches like the white mattress at the bottom. This shows that the model hasn't learned a structured or useful latent representation.

Experiment **V10.10**, The random samples at epoch 50 are completely meaningless, flat, noisy textures with no resemblance to real data. This means that skip connections give too many details directly copying features from the encoder to the decoder, reducing pressure on the latent space to learn a true representation of the data.

Experiment **V10.02** produced much better results. While samples at epoch 1 were noisy, by epoch 25, structure and colour patterns occurred, and by epoch 50, some frames showed the actual backgrounds seen in the dataset. Though the subject remained hard to visualise, the model showed a clear ability to learn spatial patterns and structure, likely due to the larger training set. This shows that training on a bigger dataset improves latent space quality, even without architectural enhancements like skip connections.

5.1.3 Loss Evaluation

Across all experiments, total and reconstruction losses show smooth convergence, with validation closely following training loss, meaning fair generalisation in all three models. However, the KL divergence curves show interesting key differences in latent space behaviour.

V10.00 (Figure 8): The KL divergence drops quickly in the early epochs and then stabilises. This suggests the model learns a consistent but relatively simple latent space. The absence of skip connections forces the model to rely on the latent representation, yet it does not fully exploit fully, likely because of the limited training data (only fall samples).

V10.10 (Figure 11): The KL loss collapses early and stays very low (under 1), which is a sign of posterior collapse. Posterior collapse occurs when the latent variables stop encoding meaningful information about the input, causing the learned posterior distribution $q(z|x)$ to collapse to the prior distribution $p(z)$. Research shows that skip connections can help prevent posterior collapse when used to support the connection between latent variables and the decoder, the skip connections used in this model actually ignore the latent space completely (Dieng et al., 2019). By passing high-resolution features from the encoder directly to the decoder, the model is able to reconstruct inputs without relying on the latent variable. This architectural shortcut leads to posterior collapse, as evidenced by the very low KL divergence and meaningless random samples observed in the experiment.

V10.02 (Figure 14): KL divergence shows frequent spikes, particularly in early training. This reflects a more complex and unstable optimisation of the latent space likely due to the diverse input (fall + non-fall).

The results of this project show that integrating skip connections into the Variational Autoencoder (VAE) architecture improves the visual quality of reconstructed video data, which aligns with Xu et al. (2024) and He et al. (2015), who highlighted the role of skip connections in preserving critical features in deep learning models.

5.2 Reconstructed Data Evaluation on Fall Detection model

To evaluate the effectiveness of the reconstructed data generated by the VideoVAE, I tried to fine-tune the existing fall detection model using the visually best reconstructed data from Experiment V10.10 (trained only on fall clips, with skip connections enabled). However, several limitations affected the evaluation process.

In order to properly compare the base model and the fine-tuned model, it would have been ideal to include multiple activity classes beyond just falls. Although I have access to the base model's architecture and trained weights, I do not have access to the original test dataset that was used to evaluate its performance. This made it impossible to run a direct evaluation or reproduce the original results.

To overcome this, I used all 1,010 original videos (including both fall and non-fall clips) from the UP-Fall Dataset and classified them using the base model.

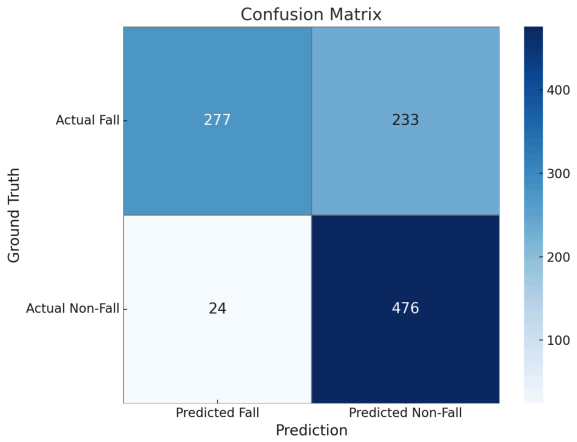


Figure 15: Confusion matrix for the base model, showing the classification results on 1,010 original fall and non-fall videos before fine-tuning.

Next, I fine-tuned the base model using the best reconstructed video data from Experiment V10.10. Since this reconstructed data only contains fall activities (one class), which resulted, no meaningful model evaluation could be performed during training, such as accuracy or recall across multiple classes.

After fine-tuning, I used the updated model to classify the same 1,010 original videos as before.

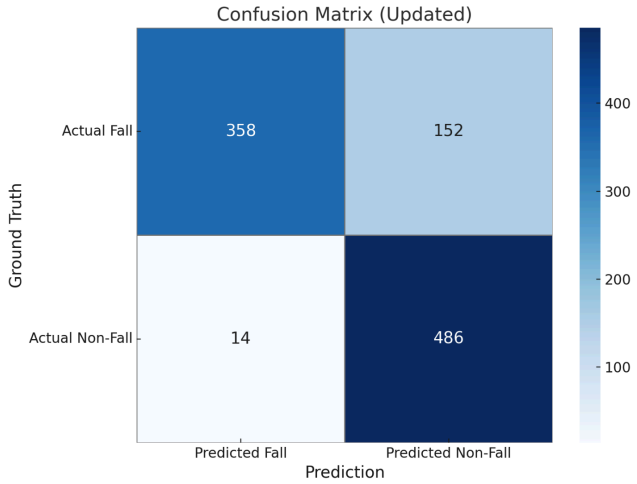


Figure 16: Confusion matrix for the fine-tuned model, showing the classification results on 1,010 original fall and non-fall videos after the fine-tuning.

I then compared the two confusion matrix. The results indicates improvement in classification performance after fine-tuning. However, this improvement cannot be taken as definitive evidence of model enhancement, because the fine-tuned model effectively saw the same data twice, the original and its reconstructed version, which may have led to overfitting. The reconstructed videos, although generated, are still visually similar to the original input, meaning the model may have memorised the patterns rather than generalised them.

	Base Model	Fine-Tuned Model	Improvements (%)
Accuracy	0.74	0.83	12.08
Precision	0.92	0.96	4.57
Recall	0.54	0.70	29.24
F1-Score	0.68	0.81	18.83

Figure 17: Confusion matrix comparison of the key classification metrics from pre and post fine-tunied model

While these results are promising, further evaluation with unseen test data and multi-class training is necessary to validate the true effectiveness of the synthetic data for model improvement.

6. Conclusions and Future Work

The primary aim of this project was to generate synthetic fall-related video data using a Variational Autoencoder (VAE) model, to increase the limited real-world data for fall detection systems. By adopting a traditional image-based VAE to support video data via 3D convolutions and experimenting with architectural modifications like skip connections, I was able to explore both the potential and limitations of VAEs for synthetic video generation.

Three key experiments were executed to understand how different variables like dataset size and network architecture can affect the model's performance. The results clearly demonstrated that skip connections can significantly improve the visual quality of reconstructions by preserving low-level spatial features. However, they also showed an important trade-off: while skip connections enhanced reconstruction, they led to posterior collapse, where the latent space fails to learn meaningful features. In contrast, the version trained on a larger dataset without skip connections showed more meaningful latent space generation, better generalisation, and more structured outputs.

Although reconstructed and synthetic data from these experiments have not yet led to performance improvements in the fall detection classifier, the project successfully achieved its goals: building a working VideoVAE model, modifying its structure to handle temporal video input, and exploring the impact of skip connections on data generation and latent space learning.
For future work, I'm going to explor:

- Latent-to-decoder skip connections: Instead of bypassing the latent space entirely, I recently discovered that it is possible to introduce skip connections *from* the latent space *to* the decoder inspired by Dieng et al. (2019), which link the latent space directly to decoder layers. Unlike traditional skip connections that bypass latent representations, the approach proposed by Dieng et al. (2019) injects the latent variable into multiple layers of the generative model. These "generative skip models" improve the dependence between the latent space and the output.
- VQ-VAE (Vector Quantised VAE): This variant can produce discrete latent representations, potentially improving sample quality and offering better control over generated data. It could also mitigate some of the issues observed with KL divergence and posterior collapse.
- Using Optical Flow as Input: Training on motion-encoded data rather than raw RGB frames could reduce background noise, allowing the model to focus more effectively on motion patterns relevant to falls. This could improve the model's ability to generate motion-specific features. This suggestion comes directly from the KTP classifier's use of optical flow and its success in reducing background noise (Toh et al., 2024)

7 Project Management Review

Looking back at my initial Gantt chart and comparing it to the final version, there were many adjustments along the way. Some tasks took longer than expected, while others had to be added later once I understood the problem better. Overall, the timeline had to be extended, especially for model evaluation and experiment tracking, which turned out to be more time, consuming than I originally planned.

One of my strengths during this project was how I structured my work environment. I used Google Colab for training and linked it with Google Drive for storing results and backups. This setup made it easy to switch between experiments and minimising the risk of avoid data loss by storing everything in the claud. I also kept my project well organised, with clear versioning and logs, which helped a lot when comparing experiments or going back to older results.

Another strength was my ability to adapt. At the beginning, I had limited knowledge of Variational Autoencoders (VAEs), but I dedicated time to gain clear understanding how VAEs works. This learning allowed me to build a working model, modify it for video input, and try different architectural improvements like skip connections.

However, there were also weaknesses. I underestimated the complexity of evaluating generative models. It took me a while to figure out how to properly assess the results, and this caused some delays.

Also, I didn't have full access to the fall detection model used in the KTP project, so I couldn't immediately test how my synthetic data affected its performance. This limited the end-to-end validation of my approach.

I also discovered some useful tools, like ThreadPoolExecutor, quite late in the project. If I had learned about it earlier, I could have saved time and reduced my usage of Colab computing units, which would have helped both my productivity and budget.

Overall, this project helped me develop both my technical skills and my ability to manage a complex task with many moving parts. I learned how to plan, adapt, troubleshoot, and keep track of different experiments. If I were to do it again, I would spend more time understanding the full pipeline from the beginning and plan for more time to evaluate results properly.

References

Chen, Z., Wang, Y. and Yang, W., 2021. *Video based fall detection using human poses*. arXiv. Available at: https://doi.org/10.48550/arXiv.2107.14633 (Accessed: October 2024).

Dieng, A.B., Kim, Y., Rush, A.M. and Blei, D.M., 2019. *Avoiding latent variable collapse with generative skip models*. In: *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 2397–2405. Available at: https://proceedings.mlr.press/v89/dieng19a/dieng19a.pdf (Accessed: February 2025).

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. *Generative adversarial nets*. *Advances in Neural Information Processing Systems*, 27. Available at: https://papers.nips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf (Accessed: October 2024).

He, K., Zhang, X., Ren, S. and Sun, J., 2015. *Deep residual learning for image recognition*. arXiv. Available at: https://arxiv.org/abs/1512.03385 (Accessed: January 2025).

Kingma, D.P. and Welling, M., 2013. *Auto-encoding variational Bayes*. arXiv. Available at: https://doi.org/10.48550/arXiv.1312.6114 (Accessed: September 2024).

Lia, C., Zhang, T., Du, X., Zhang, Y. and Xie, H., 2024. *Generative AI models for different steps in architectural design: A literature review*. arXiv. Available at: https://arxiv.org/abs/2404.01335v2 (Accessed: November 2024).

Lu, M., Shen, M. and Zhang, J., 2018. *A probe towards understanding GAN and VAE models*. arXiv. Available at: https://doi.org/10.48550/arXiv.1812.05676 (Accessed: November 2024).

Lu, Y. et al., 2024. *Machine learning for synthetic data generation: A review*. arXiv. Available at: https://arxiv.org/abs/2302.04062v9 (Accessed: November 2024).

Martínez-Villaseñor, L., Ponce, H., Brieva, J., Moya-Albor, E., Núñez-Martínez, J. and Peñafort-Asturiano, C., 2019. *UP-Fall Detection Dataset: A multimodal approach*. *Sensors*, 19(9), p.1988. Available at: https://www.mdpi.com/1424-8220/19/9/1988 (Accessed: September 2024).

Nikolentzos, G., Vazirgiannis, M., Xypolopoulos, C., Lingman, M. and Brandt, E.G., 2023. *Synthetic electronic health records generated with variational graph autoencoders*. *NPJ Digital Medicine*, 6, p.83. Available at: https://www.nature.com/articles/s41746-023-00822-x (Accessed: November 2024).

Stephenson Harwood, 2024. *AI, machine learning and big data laws and regulations 2024*. [online] Available at: https://www.shlegal.com/insights/ai-machine-learning-and-big-data-laws-and-regulations-2024 (Accessed: February 2025).

Subramanian, A.K., 2020. *PyTorch-VAE*. GitHub repository. Available at: https://github.com/AntixK/PyTorch-VAE (Accessed: November 2024).

Toh, S.M., Helian, N., Pasipamire, K., Sun, Y. and Pasipamire, T., 2024. *Vision-based human fall detection using 3D neural networks*. Available at: https://link.springer.com/chapter/10.1007/978-3-031-77918-3_4 (Accessed: September 2024).

Xu, G., Wang, X., Wu, X., Leng, X. and Xu, Y., 2024. *Development of skip connection in deep neural networks for computer vision and medical image analysis: A survey*. arXiv. Available at: https://arxiv.org/abs/2405.01725v1 (Accessed: February 2025).

Zbeeb, M., Ghorayeb, M. and Salman, M., 2024. *Exploring the landscape for generative sequence models for specialized data synthesis*. arXiv. Available at: https://arxiv.org/abs/2411.01929v2 (Accessed: February 2025).

Bibliography

Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep Learning*. Cambridge, MA: MIT Press.

Appendices

A - Repositories and Data Sources

- **My Project GitHub Repository:** *Contains all code, models, configurations, and training logs related to this project with instructions.*
  - https://github.com/Acidias/Final-Year-Project
- **Processed VideoData:** The .npy file, used to train the VideoVAE
  - Compressed: https://drive.google.com/file/d/1dsFVObf37dOQimVAiLsQP43A-g7A0TTd/view?usp=sharing
  - Base NPY File: https://drive.google.com/file/d/1tyTjevVW04GuAczbzuVk68PYEInOKWxC/view?usp=sharing
- **Experiment v10.00:** https://drive.google.com/drive/folders/1UFfv8oN8Emx0qKxHg4wXMGCpgNx3QD-2?usp=sharing
- **Experiment v10.02:** https://drive.google.com/drive/folders/1_ZKLH9qeT4jV87ryFa_02GerWkbjOdy_?usp=sharing
- **Experiment v10.10:** https://drive.google.com/drive/folders/165292UNObAkhtlDzdx1bl3KzNiB9G_l7?usp=sharing
- **Base VAE Repository:** *Original PyTorch VAE architecture used as a starting point for modifications in this project.*
  - https://github.com/AntixK/PyTorch-VAE
- **UP-Fall Detection Dataset:** *Dataset used in the project for training and evaluating fall detection models and the VideoVAE.*
  - https://www.researchgate.net/publication/332730828_UP-Fall_Detection_Dataset_A_Multimodal_Approach

B - Figure References

- **Figure 2 – Illustration of an Autoencoder with Skip Connections:**
  - https://sensibilityit.tistory.com/517
- **Figure 3 – UP-Fall Detection Activity Descriptions:**
  - https://www.researchgate.net/publication/332730828_UP-Fall_Detection_Dataset_A_Multimodal_Approach